Volume 6, Issue 5, May 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

# An Efficient and Reliable Key Swapping Technique Designed for Parallel Network File Systems

**K. Soundararajan**
M.E., Department of Computer Science,
Staff Member of Vivekanandha College of Tech for Women,
Sathinaickanpalayam, Tiruchengode, Tamilnadu, India

**S. Malathi**
M.E., Department of Computer Science
Student of Vivekanandha College of Tech. for Women,
Sathinaickanpalayam, Tiruchengode, Tamilnadu, India

***Abstract— Work focuses on efficiency and scalability with respect to the metadata server. The problem of secure many-to-many communications in large-scale network file systems that support parallel access to multiple storage devices. It is considered as 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. The goal of this project is to reduce the workload of the metadata server. The computational and communication overhead for both the client and the storage device should remain reasonably low. The current Internet standard for file systems, like Parallel Network File System which makes use of Kerberos establish parallel session keys between clients and storage devices. The Kerberos-based protocol shows that it has a number of limitations and proposes a variety of authenticated key exchange protocols that are designed to address the above issues. The idea is to demonstrate that the protocols are competent of dropping up to approximately 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. The main technique used here is ECDH which is an anonymous key exchange protocol that allows two parties, each having an elliptic curve public–private key pair, to establish a shared secret over an insecure channel.***

***Keywords— Kerberos, Authenticated key exchange, Metadata server, forward secrecy, escrow-freeness.***

## 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 *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), 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. 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 environ-ments, 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.

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:

- *Scalability* – the metadata server facilitating access re-quests 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.

Boneh D, Gentry C and Waters B [1] has describes two new public key broadcast encryption systems for stateless receivers. Both systems are fully secure against any number of colluders. In first construction both cipher texts and private keys are of constant size (only two group elements), for any subset of receivers. The public key size in this system is linear in the total number of receivers. The second system is a generalization of the first that provides a trade-off between cipher text size and public key size. For example, A collusion resistant broadcast system for n users where both cipher texts and public keys are of size O(pn) for any subset of receivers.

Canetti R and Krawczyk H [2] Present 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:

- Any key-exchange protocol that satisfies the security definition can be composed with symmetric encryption and authentication functions to provide provably secure communication channels.
- 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 those using general tools to obtain security in the realistic setting of adversary-controlled links.

Leung A W, Miller E L and Jones S [3] said that Petascale, high-performance file systems often hold sensitive data and thus require security, but authentication and authorization can dramatically reduce performance. Existing security solutions perform poorly in these environments because they cannot scale with the number of nodes, highly distributed data, and demanding workloads. To address these issues, they developed Maat, a security protocol designed to provide strong, scalable security to these systems. Maat introduces three new techniques. Access control is the primary contributor to security overhead because the number of capabilities and their resulting cryptographic overhead tends to scale up as systems or workloads become larger.

Leung A W and Miller E L [4] Present a designs for petabyte-scale storage systems are now capable of transferring hundreds of gigabytes of data per second, but lack strong security. A scalable and efficient protocol for security in high performance, object based storage systems that reduces protocol overhead and eliminates bottlenecks, thus increasing performance without sacrificing security primitives. The protocol enforces security using cryptographically secure capabilities, with three novel features that make them ideal for high performance workloads: a scheme for managing coarse grained capabilities, methods for describing client and file groups, and strict security control through capability lifetime extensions. By reducing the number of unique capabilities that must be generated, metadata server load is reduced.

White B S, Walker M, Humphrey M and Grimshaw M [5] Realizing that current file systems cannot cope with the diverse requirements of wide-area collaborations, researchers have developed data access facilities to meet their needs. Recent work has focused on comprehensive data access architectures. In order to fulfill the evolving requirements in this environment, it suggests a more fully-integrated architecture built upon the fundamental tenets of naming, security, scalability, extensibility, and adaptability. These form the underpinning of the Legion File System. This paper motivates the need for these requirements and presents benchmarks that highlight the scalability of Legion FS. The serverless architecture of Legion FS is shown to benefit important scientific applications, such as those accessing the Protein Data Bank, within both local- and wide-area environments.

## II. INTERNET STANDARD — NFS

NFS was developed by Sun Microsystems and has been designated a file server standard. Its protocol uses the Remote Procedure Call (RPC) method of communication between computers. You can install NFS on Windows 95 and some other operating systems using products like Sun's Solstice Network Client. Using NFS, the user or a system administrator can mount all or a portion of a file system (which is a portion of the hierarchical tree in any file directory and subdirectory, including the one you find on your PC or Mac).

The portion of your file system that is mounted (designated as accessible) can be accessed with whatever privileges go with your access to each file (read-only or read-write). NFS has been extended to the Internet with WebNFS, a product and proposed standard that is now part of Netscape's Communicator browser. WebNFS offers what Sun believes is a faster way to access Web pages and other Internet files.

### *A. Current Limitations*

The current design of NFS/pNFS focuses on *interoperability*, instead of efficiency and scalability, of various mechanisms to provide basic security. Moreover, key establishment between a client and multiple storage devices in pNFS are based on those for NFS, that is, they are not designed specifically for parallel communications. Hence, the metadata server is not only responsible for processing access requests to storage de-vices (by granting valid layouts to authenticated and authorized clients), but also required to generate all the corresponding session keys that the client needs to communicate securely with the storage devices to which it has been granted access. Consequently, the metadata server may become a performance bottleneck for the file system. Moreover, such protocol design leads to key escrow. Hence, in principle, the server can learn all information transmitted between a client and a storage device. This, in turn, makes the server an attractive target for attackers.

Another drawback of the current approach is that past session keys can be exposed if a storage device's long-term key shared with the metadata server is compromised. We believe that this is a realistic threat since a large-scale file system may have thousands of geographically distributed storage devices. It may not be feasible to provide strong physical security and network protection for all the storage devices.

## III. ELLIPTIC CURVE CRYPTOGRAPHY

Elliptic Curve Cryptography (ECC) was first proposed by victor Miller and independently by Neal Koblitz in the mid-1980s and has evolved into a mature public-key cryptosystem. Compared to its traditional counterparts, ECC offers the same level of security using much smaller keys. This result in faster computations and savings in memory, power and bandwidth those are especially important in constrained environments. More significantly, the advantage of ECC over its competitor's increases, as the security needs increase over time. The Elliptic curve cryptography (ECC) is an emerging favorite because requires less computational power, communication bandwidth, and memory when compared to other cryptosystems.

### *A. Key establishment protocol*

The elliptic curve algorithms will work in a cyclic subgroup of an elliptic curve over a finite field. The algorithms will need the following parameters, the prime $p$ that specifies the size of the finite field. The coefficients $a$ and $b$ of the elliptic curve equation. The base point $G$ that generates our subgroup. The order $n$ of the subgroup. The cofactor $h$ of the subgroup.

In conclusion, the domain parameters for our algorithms are the six tuple $(p,a,b,G,n,h)$.

1. The private key is a random integer $d$ chosen from $\{1,\ldots,n-1\}$ (where $n$ is the order of the subgroup).
2. The public key is the point $H=dG$ (where $G$ is the base point of the subgroup).

An elliptic curve is a plane curve defined by an equation of the form
$y^2 = x^3 + ax + b$ (1)
where $a$ and $b$ are real numbers. The curve $y^2 = x^3 - px - q$ over the field $K$ and points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ on the curve, assume first that $x_P \neq x_Q$. Let $s$ be the slope of the line containing $P$ and $Q$; i.e.,
$s = \frac{y_P - y_Q}{x_P - x_Q}$ (2)
Since $K$ is a field, $s$ is well-defined. Then we can define $R = P + Q = (x_R - y_R)$ by
$x_R = s^2 - x_P - x_Q$ (3)
$y_R = y_P + s(x_R - x_P)$ (4)
If $x_P = x_Q$ (third and fourth panes below), then there are two options: if $y_P = -y_Q$, including the case where $y_P = y_Q = 0$, then the sum is defined as 0; thus, the inverse of each point on the curve is found by reflecting it across the $x$-axis. If $y_P = y_Q \neq 0$, then $R = P + P = 2P = (x_R - y_R)$ is given by
$s = \frac{3x_P{}^2 - p}{2y_P}$ (5)
$x_R = s^2 - 2x_P$ (6)
$y_R = y_P + s(x_R - x_P)$ (7)

All of the group laws except associativity follow immediately from the geometrical definition of the group operation. This animation illustrates geometrically the associativity law.

## IV. DESCRIPTION OF OUR PROTOCOLS

Our protocols, progressively designed to achieve efficiency and security. Show that the 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. An appropriate security model and prove that our protocols are secure in the model.

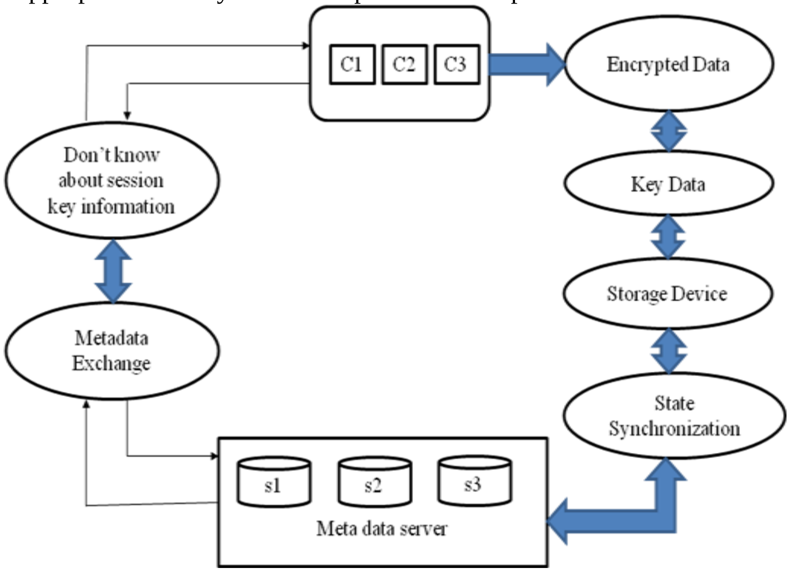

Fig.1. The concept model of pNFS.

### *A. 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. CONCLUSION

The ECDH (Elliptic curve Diffie–Hellman) is used for key exchange. ECDH is an anonymous key exchange protocol that allows two parties, each having an elliptic curve public–private key pair, to establish a shared secret over an insecure channel. First, the metadata server executing our protocols has much lower workload than that of the Kerberos-based approach. 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. Second, two our protocols provide forward secrecy: one is partially forward secure, 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. 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.

## REFERENCES

[1] Boneh D, Gentry C and Waters B (2005), ‘Collusion Resistant Broadcast Encryption with Short Cipher Texts and Private Keys’. In Advances in Cryptology – Proceedings of CRYPTO, pages 258–275. Springer LNCS 3621.

[2] 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, pages 453–474. Springer LNCS 2045.

[3] Kubiatowicz J, Bindel D, Chen Y, Czerwinski S E, Eaton P R, Geels Gummadi R, Rhea S C, Weatherspoon H, Weimer W, Wells and Zhao B Y (2000), ‘OceanStore: An architecture for Global-scale Persistent Storage’. In Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pages 190–201. ACM Press.

[4] Leung A W, Miller E L and Jones S (2007), ‘Scalable Security for Petascale Parallel File Systems’. In Proceedings of the ACM/IEEE Conference on High Performance Networking and Computing (SC), page 16. ACM Press.

[5] Leung A W and Miller E L (2006), ‘Scalable Security for large, high Performance Storage Systems’. In Proceedings of the ACM Workshop on Storage Security and Survivability (Storage SS), pages 29–40. ACM Press.

[6] Mazieres D, Kaminsky M, Kaashoek M F and Witchel E (1999), ‘Separating Key Management from File System Security’. In Proceedings of the 17th ACM Symposium on Operating System Principles (SOSP), pages 124–139.ACM Press.

[7] Mell P and Grance T (2011), ‘The NIST definition of Cloud Computing’. National Institute of Standards and Technology (NIST), Special Publication 800-145.

[8] Olson C and Miller E L (2005), ‘Secure Capabilities for a Petabyte-scale Object-based Distributed File System’. In Proceedings of the ACM Workshop on Storage Security and Survivability (StorageSS), pages 64–73. ACM Press.

[9] O’Malley O, Zhang K, Radia S, Marti R and Harrell C (2009), ‘Hadoop Security Design’. Yahoo!, https://issues.apache.org/jira/secure/ attachment/12428537/security-design.pdf.

[10] Shepler S, Eisler M and Noveck D (2010), ‘Network File System (NFS) version 4 minor version 1 protocol’. The Internet Engineering Task Force (IETF), RFC 5661.

[11] Weil S A and Miller E L, Long D D E and Maltzahn C (2006), ‘Ceph: A Scalable, High-performance Distributed File System’. In Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI), pages 307–320. USENIX Association.

[12] White B S, Walker M, Humphrey M and Grimshaw M (2001), ‘A.S. LegionFS: A Secure and Scalable File System supporting cross-domain high Performance Applications. In Proceeding of the ACM/IEEE Conference on Supercomputing (SC), page 59. ACM Press.

[13] Zhu Y. and HuY. (2003), ‘SNARE: A strong security scheme for network-attached storage’. In Proceedings of the 22nd Symposium on Reliable Distributed Systems (SRDS), pages 250–259. IEEE Computer Society.