CHAPTER 7

CONCLUSION

In this work, a secure, scalable and efficient model for cloud store is proposed and implemented on top of HBase. The goal is to provide an improved level of security for cloud storage which is one of the major concerns of data deployed in cloud in recent years. This model is designed primarily with the aim of securing data at rest using Map-Reduce. The evaluation proved that cloud data is relatively more secure with AES+XTS mode when compared to AES+EBC mode. The findings prove that AES+XTS followed by compression under mapper nodes is a secure parallel technique for large dataset. The results prove that the encryption process consumed less time in the order of 12 minutes for 5 Gb under mapper alone. The findings in chapter 2 give rise to 4 issues viz:

1) When data is processed under Map-Reduce paradigm, if the security of the algorithm is not ensured, it is meaningless to use the parallel structure for processing large data. Therefore Map-Reduce process should also be secured and hence the key encryption technique has been proposed to impart security of the parallel algorithm. Specific tokens are introduced such as BlockAccessTokens to access blocks and JobTokens for running jobs under DataNodes. The experimental result of the proposed techniques show that the key encryption technique requires lesstime in the order of milliseconds for a 32 node cluster.
2) The encrypted data left in cloud store needs privacy, which is resolved in this thesis by using the homomorphic property of cryptographic algorithms. The multiplicative and additive homomorphic property of RSA and Pailler cryptosystem running under Map-Reduce model, consumed relatively less time in the order of seconds. This technique is efficient when compared to the proposed error detection scheme for odd and even chunks. The property checks allow identifying whether a fault has been injected in the encrypted data. The experimental results prove that the additive homomorphism (Pailler System) consumed more time than the multiplicative homomorphism (RSA System).

3) The third issue identified with respect to cloud store is that when the same data is getting deployed in the cloud it consumes more CPU cycles by running the AES+XTS followed by compression on similar data deployed in the cloud. Deduplication is a technique used to resolve the issue. The Rabin-Karp algorithm used under different polynomial sizes Rabin-32 and Rabin-54 seems to give an improved deduplication ratio in the order of 1:400.

4) Without a scalable environment, it is hard to discuss about processing large data in the cloud. Therefore the last chapter of this thesis explains how scalability is achieved to support the cloud environment. Scalability is achieved by the Bucket System built on top of HBase. The key-value store perform faster read and write operations when compared to other column-oriented stores. This framework is scalable to execute 40k read and write operations using HBase in seconds. This framework is finally tested against the PHR case study with
current Amazon Web Service. The PHR system proved that the HBase framework took very less time for Dictionary, Index and Bucket creation. The algorithms used for retrieving the query results prove that multi-keyword search algorithm using vector model yields better accuracy than using it with relevant score mechanism.

Future enhancements of the work involve preserving the privacy of encrypted store using Oblivious Random Access Memory (ORAM). Despite developing scalable, secure and efficient framework for cloud users still distrust the cloud provider. Even if data is encrypted, user access patterns can leak important information about the content. This is resolved using ORAM. The goal of ORAM is to completely hide the data access pattern (which blocks were read/written) from the server. Each data read or write request will generate a completely random sequence of data accesses from the server's perspective. An adversary can observe the physical storage locations accessed, but the ORAM algorithm ensures that the adversary has negligible probability of learning anything about the true (logical) access pattern. So, the future enhancement of the framework incorporates running an ORAM algorithm in cloud store to hide the data access pattern thereby further imparting privacy and security.