**2. LITERATURE SURVEY**

Literature [survey](http://www.blurtit.com/q876299.html) is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy n company strength. Once these things are satisfied, ten next steps are to determine which operating system and language can be used for developing the tool. Once the [programmers](http://www.blurtit.com/q876299.html) start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from [book](http://www.blurtit.com/q876299.html) or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

**“SPANStore: Cost-effective Geo-replicated Storage Spanning Multiple Cloud Services,”**

Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha

By offering storage services in several geographically distributed data centers, cloud computing platforms enable applications to offer low latency access to user data. However, application developers are left to deal with the complexities associated with choosing the storage services at which any object is replicated and maintaining consistency across these replicas. In this paper, we present SPANStore, a key-value store that exports a unified view of storage services in geographically distributed data centers. To minimize an application provider’s cost, we combine three key principles. First, SPANStore spans multiple cloud providers to increase the geographical density of data centers and to minimize cost by exploiting pricing discrepancies across p+roviders. Second, by estimating application workload at the right granularity, SPANStore judiciously trades off greater geo-distributed replication necessary to satisfy latency goals with the higher storage and data propagation costs this entails in order to satisfy fault tolerance and consistency requirements. Finally, SPANStore minimizes the use of compute resources to implement tasks such as two-phase locking and data propagation, which are necessary to offer a global view of the storage services that it builds upon. Our evaluation of SPANStore shows that it can lower costs by over 10x in several scenarios, in comparison with alternative solutions that either use a single storage provider or replicate every object to every data center from which it is accessed.

**“Robust Data Sharing with Key-value Stores,”**

C. Basescu, C. Cachin, I. Eyal, R. Haas, and M. Vukolic

A key-value store (KVS) offers functions for storing and retrieving values associated with unique keys. KVSs have become widely used as shared storage solutions for Internet-scale distributed applications. We present a fault-tolerant wait-free efficient algorithm that emulates a multi-reader multi-writer register from a set of KVS replicas in an asynchronous environment. Our implementation serves an unbounded number of clients that use the storage. It tolerates crashes of a minority of the KVSs and crashes of any number of clients. We provide two variants of our algorithm: one implementing an atomic register and one implementing a regular register; the latter does not require read operations to store data at the underlying KVSs. We note that applying state-of-the-art reliable storage solutions to this scenario is either impossible or prohibitively inefficient.

**“Secret-sharing schemes: A survey,” in International Workshop on Coding and Cryptology (IWCC),**

A secret-sharing scheme is a method by which a dealer distributes shares to parties such that only authorized subsets of parties can reconstruct the secret. Secret-sharing schemes are an important tool in cryptography and they are used as a building box in many secure protocols, e.g., general protocol for multiparty computation, Byzantine agreement, threshold cryptography, access control, attribute-based encryption, and generalized oblivious transfer. In this survey, we describe the most important constructions of secretsharing schemes; in particular, we explain the connections between secretsharing schemes and monotone formulae and monotone span programs. We then discuss the main problem with known secret-sharing schemes – the large share size, which is exponential in the number of parties. We conjecture that this is unavoidable. We present the known lower bounds on the share size. These lower bounds are fairly weak and there is a big gap between the lower and upper bounds. For linear secret-sharing schemes, which is a class of schemes based on linear algebra that contains most known schemes, super-polynomial lower bounds on the share size are known. We describe the proofs of these lower bounds. We also present two results connecting secret-sharing schemes for a Hamiltonian access structure to the NP vs. coNP problem and to a major open problem in cryptography – constructing oblivious-transfer protocols from one-way functions.

**“DepSky: Dependable and Secure Storage in a Cloud-ofclouds,”**

A.Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa,

The increasing popularity of cloud storage services has lead companies that handle critical data to think about using these services for their storage needs. Medical record databases, power system historical information and financial data are some examples of critical data that could be moved to the cloud. However, the reliability and security of data stored in the cloud still remain major concerns. In this paper we present DEPSKY, a system that improves the availability, integrity and confidentiality of information stored in the cloud through the encryption, encoding and replication of the data on diverse clouds that form a cloud-of-clouds. We deployed our system using four commercial clouds and used PlanetLab to run clients accessing the service from different countries. We observed that our protocols improved the perceived availability and, in most cases, the access latency when compared with cloud providers individually. Moreover, the monetary costs of using DEPSKY on this scenario is twice the cost of using a single cloud, which is optimal and seems to be a reasonable cost, given the benefits.

**“Deniable encryption with negligible detection probability: An interactive construction,”**

M. Dürmuth and D. M. Freeman

Deniable encryption, introduced in 1997 by Canetti, Dwork, Naor, and Ostrovsky, guarantees that the sender or the receiver of a secret message is able to “fake” the message encrypted in a specific ciphertext in the presence of a coercing adversary, without the adversary detecting that he was not given the real message. To date, constructions are only known either for weakened variants with separate “honest” and “dishonest” encryption algorithms, or for single-algorithm schemes with non-negligible detection probability.

We propose the first sender-deniable public key encryption system with a single encryption algorithm and negligible detection probability. We describe a generic interactive construction based on a public key bit encryption scheme that has certain properties, and we give two examples of encryption schemes with these properties, one based on the quadratic residuosity assumption and the other on trapdoor permutations.

**“AONT-RS: Blending Security and Performance in Dispersed Storage Systems,”**

J. K. Resch and J. S. Plank,

Dispersing files across multiple sites yields a variety of obvious benefits, such as availability, proximity and reliability. Less obviously, it enables security to be achieved without relying on encryption keys. Standard approaches to dispersal either achieve very high security with correspondingly high computational and storage costs, or low security with lower costs. In this paper, we describe a new dispersal scheme, called AONT-RS, which blends an All-Or-Nothing Transform with Reed-Solomon coding to achieve high security with low computational and storage costs. We evaluate this scheme both theoretically and as implemented with standard open source tools. AONTRS forms the backbone of a commercial dispersed storage system, which we briefly describe and then use as a further experimental testbed. We conclude with details of actual deployments.