INTERNATIONAL JOURNAL FOR DEVELOPMENT OF COMPUTER SCIENCE & TECHNOLOGY
VOLUME-1, ISSUE-III (April-May 2013) IS NOW AVAILABLE AT: www.ijdcst.com
ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

Enhanced Data Protection in Cloud through Encryption Algorithms

[1]Sunil Kumar Mangi , [2]G.Guru Kesava Dasu
[1]M.Tech Student, [2]Professor, H.O.D (C.S.E &IT)
[1,2]Department of Computer Science and Electronics
[1,2]Eluru College Of Engineering & Technology, Eluru

Abstract— Cloud Computing has been envisioned as the next generation architecture of IT Enterprise. In contrast to traditional solutions, where the IT services are under proper physical, logical and personnel controls, Cloud Computing moves the application software and databases to the large data centers, where the management of the data and services may not be fully trustworthy. This unique attribute, however, poses many new security challenges which have not been well understood. In this article, we focus on cloud data storage security, which has always been an important aspect of quality of service. The data protection-as-a-service cloud platform architecture dramatically reduces the per-application development effort required to offer data protection while still allowing rapid development and maintenance In this paper The Diffie Hellman key exchange algorithm is used for proving Data Production in clouds.

Keywords— Cloud Computing, Encryption, Services, SAAS.

I. Introduction

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation.

In the business model using software as a service, users are provided access to application software and databases. The cloud providers manage the infrastructure and platforms on which the applications run. SaaS is sometimes referred to as "on-demand software" and is usually priced on a pay-per-use basis. SaaS providers generally price applications using a subscription fee.

Proponents claim that the SaaS allows a business the potential to reduce IT operational costs by outsourcing hardware and software maintenance and support to the cloud provider. This enables the business to reallocate IT operations costs away from hardware/software spending and personnel expenses, towards meeting other IT goals. In addition, with applications hosted centrally, updates can be released without the need for users to install new software. One drawback of SaaS is that the users' data are stored on the cloud provider's server. As a result, there could be unauthorized access to the data.

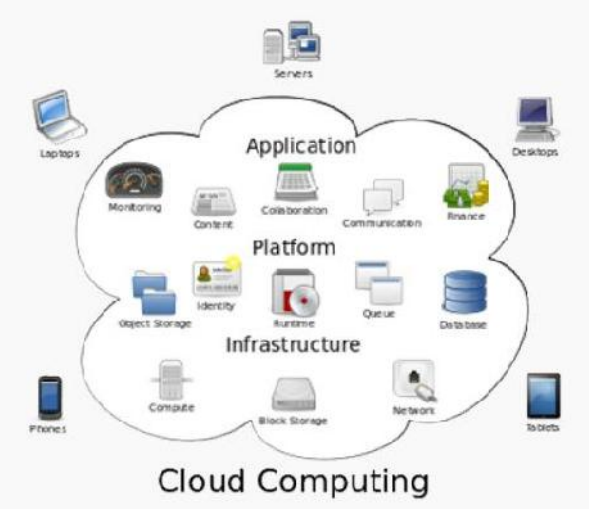

Fig. 1.Cloud computing logical diagram

End users access cloud-based applications through a web browser or a light-weight desktop or mobile app while the business software and user's data are stored on servers at a remote location. Proponents claim that cloud computing allows enterprises to get their applications

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

up and running faster, with improved manageability and less maintenance, and enables IT to more rapidly adjust resources to meet fluctuating and unpredictable business demand.

Cloud computing relies on sharing of resources to achieve coherence and economies of scale similar to a utility (like the electricity grid) over a network. At the foundation of cloud computing is the broader concept of converged infrastructure and shared services.

Several trends are opening up the era of Cloud Computing, which is an Internet-based development and use of computer technology. The ever cheaper and more powerful processors, together with the software as a service (SaaS) computing architecture, are transforming data centers into pools of computing service on a huge scale. The increasing network bandwidth and reliable yet flexible network connections make it even possible that users can now subscribe high quality services from data and software that reside solely on remote data centers. Moving data into the cloud offers great convenience to users since they don't have to care about the complexities of direct hardware management. The pioneer of Cloud Computing vendors, Amazon Simple Storage Service (S3) and Amazon Elastic Compute Cloud (EC2) are both well known examples. While these internet-based online services do provide huge amounts of storage space and customizable computing resources, this computing platform shift, however, is eliminating the responsibility of local machines for data maintenance at the same time. As a result, users are at the mercy of their cloud service providers for the availability and integrity of their data. Recent downtime of Amazon's S3 is such an example.

From the perspective of data security, which has always been an important aspect of quality of service, Cloud Computing inevitably poses new challenging security threats for number of reasons. Firstly, traditional cryptographic primitives for the purpose of data security protection can not be directly adopted due to the users' loss control of data under Cloud Computing. Therefore, verification of correct data storage in the cloud must be conducted without explicit knowledge of the whole data. Considering various kinds of data for each user stored in the cloud and the demand of long term continuous assurance of their data safety, the problem of verifying correctness of data storage in the cloud becomes even more challenging.

Secondly, Cloud Computing is not just a third party data warehouse. The data stored in the cloud may be frequently updated by the users, including insertion, deletion, modification, appending, reordering, etc. To ensure storage correctness under dynamic data update is hence of paramount importance. However, this dynamic feature also makes traditional integrity insurance techniques futile and entails new solutions. Last but not the least, the deployment of Cloud Computing is powered by data centers running in a simultaneous, cooperated and distributed manner. Individual user's data is redundantly stored in multiple physical locations to further reduce the data integrity threats. Therefore, distributed protocols for storage correctness assurance will be of most importance in achieving a robust and secure cloud data storage system in the real world. However, such important area remains to be fully explored in the literature.

II. PROBLEM STATEMENT

A. *System Model*

Representative network architecture for cloud data storage is illustrated in Figure 1. Three different network entities can be identified as follows:

• *User:* users, who have data to be stored in the cloud and rely on the cloud for data computation, consist of both individual consumers and organizations.

• *Cloud Service Provider (CSP):* a CSP, who has significant resources and expertise in building and managing distributed cloud storage servers, owns and operates live Cloud Computing systems.

• *Third Party Auditor (TPA):* an optional TPA, who has expertise and capabilities that users may not have, is trusted to assess and expose risk of cloud storage services on behalf of the users upon request.

In cloud data storage, a user stores his data through a CSP into a set of cloud servers, which are running in a simultaneous, cooperated and distributed manner. Data redundancy can be employed with technique of erasure-correcting code to further tolerate faults or server crash as user's data grows in size and importance. Thereafter, for application purposes, the user interacts with the cloud servers via CSP to access or retrieve his data. In some cases, the user may need to perform block level operations on his data.

The most general forms of these operations we are considering are block update, delete, insert and append. As

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

users no longer possess their data locally, it is of critical importance to assure users that their data are being correctly stored and maintained. That is, users should be equipped with security means so that they can make continuous correctness assurance of their stored data even without the existence of local copies.

In case that users do not necessarily have the time, feasibility or resources to monitor their data, they can delegate the tasks to an optional trusted TPA of their respective choices. In our model, we assume that the point-to-point communication channels between each cloud server and the user is authenticated and reliable, which can be achieved in practice with little overhead. Note that we don't address the issue of data privacy in this paper, as in Cloud Computing, data privacy is orthogonal to the problem we study here.

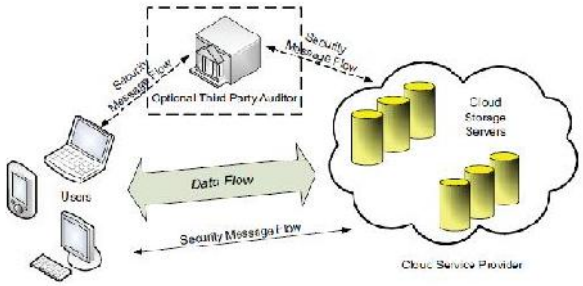

Fig 2 Cloud Data Storage Architecture

III. CLOUD COMPUTING SECURITY

Cloud computing security (sometimes referred to simply as "cloud security") is an evolving sub-domain of computer security, network security, and, more broadly, information security. It refers to a broad set of policies, technologies, and controls deployed to protect data, applications, and the associated infrastructure of cloud computing.

Cloud security is not to be confused with security software offerings that are "cloud-based" (a.k.a. security-as-a-service).

A) Security issues associated with the cloud

There are a number of security issues/concerns associated with cloud computing but these issues fall into two broad categories: Security issues faced by cloud providers and security issues faced by their customers. In most cases, the provider must ensure that their infrastructure is secure and that their clients' data and applications are protected while the customer must ensure that the provider has taken the proper security measures to protect their information.

The extensive use of virtualization in implementing cloud infrastructure brings unique security concerns for customers or tenants of a public cloud service.[3] Virtualization alters the relationship between the OS and underlying hardware - be it computing, storage or even networking. This introduces an additional layer - virtualization - that itself must be properly configured, managed and secured.[4] Specific concerns include the potential to compromise the virtualization software, or "hypervisor". While these concerns are largely theoretical, they do exist.[5] For example, a breach in the administrator workstation with the management software of virtualization software can cause whole datacenter to go down or reconfigured to attacker's liking.

Cloud Security Controls

Cloud security architecture is only effective if the correct defensive implementations are in place. An efficient cloud security architecture should recognize the issues that will arise with security management. The security management addresses these issues with security controls. These controls are put in place to safeguard any weaknesses in the system and reduce the effect of an attack. While there are many types of controls behind a cloud security architecture, they can usually be found in one of the following categories:

a) Deterrent Controls

These controls are set in place to prevent any purposeful attack on a cloud system. Much like a warning sign on a fence or a property, these controls do not reduce the actual vulnerability of a system.

b) Preventative Controls

These controls upgrade the strength of the system by managing the vulnerabilities. The preventative control will safeguard vulnerabilities of the system. If an attack were to occur, the preventative controls are in place to cover the attack and reduce the damage and violation to the system's security.

c) Corrective Controls

Corrective controls are used to reduce the effect of an attack. Unlike the preventative controls, the corrective controls take action as an attack is occurring.

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

d) Detective Controls

Detective controls are used to detect any attacks that may be occurring to the system. In the event of an attack, the detective control will signal the preventative or corrective controls to address the issue.

Security and Privacy

Identity management

Every enterprise will have its own identity management system to control access to information and computing resources. Cloud providers either integrate the customer's identity management system into their own infrastructure, using federation or SSO technology, or provide an identity management solution of their own.

Physical and personnel security

Providers ensure that physical machines are adequately secure and that access to these machines as well as all relevant customer data is not only restricted but that access is documented.

Availability

Cloud providers assure customers that they will have regular and predictable access to their data and applications.

Application security

Cloud providers ensure that applications available as a service via the cloud are secure by implementing testing and acceptance procedures for outsourced or packaged application code. It also requires application security measures be in place in the production environment.

Privacy

Finally, providers ensure that all critical data (credit card numbers, for example)are masked and that only authorized users have access to data in its entirety. Moreover, digital identities and credentials must be protected as should any data that the provider collects or produces about customer activity in the cloud.

Legal issues

In addition, providers and customers must consider legal issues, such as Contracts and E-Discovery, and the related laws, which may vary by country.

1) Solution Description

We propose a new cloud computing paradigm, *data protection as a service* (DPaaS) is a suite of security primitives offered by a cloud platform, which enforces data security and privacy and offers evidence of privacy to data owners, even in the presence of potentially compromised or malicious applications. Such as secure data using encryption, logging, key management.

In an OS, processes and files are the primary units of access control, and the OS provides suitable isolation for them. Applications can do what they like within these boundaries. In a cloud setting, the unit of access control is typically a sharable piece of user data—for example, a document in a collaborative editor. Ideally, the system offers some analogous confinement of that data, restricting its visibility only to authorized users and applications while allowing broad latitude for what operations are done on it. This can make writing secure systems easier for programmers because confinement makes it more difficult for buggy code to leak data or for compromised code to grant unauthorized access to data. A malicious program might find different ways to exfiltrate data, such as employing a side channel or covert channel, but the priority here is to sup-port benign developers, while making all applications and their actions on users' sensitive data more easily auditable to catch improper usage.

One of the main concerns people and organizations have about putting data in the cloud is that they don't know what happens to it. Having a clear audit trail of when data is accessed—and by whom or what—bolsters confidence that data is being handled appropriately. Confinement can be effective for most normal user accesses, but administrative access that's outside the normal flow of user access and involves human administrators (for example, for debugging and analysis) can especially benefit from auditing.

a) *Verifiable platform support*

Bugs need to be fixed. Data needs to be updated and migrated as schemas change. Offline computation is valuable for data aggregation across users or for pre-computation of expensive functions. To reduce the risk of unaudited backdoor access, all these functions should be subject to the same authorization flows and platform-level checks as normal requests, albeit with a separate, appropriate policy.

Platform providers should build support for confinement

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

and auditing into the platform in a verifiable way. This approval has many advantages:

- application developers don't have to reinvent the wheel;
- application code is independent of ACL enforcement;
- third-party auditing and standards compliance are easier; and
- the verifiable platform extends to virtualized environments built atop it.

Finally, the cost of examining the platform is amortized across all its users, which means significant economies of scale for a large-scale platform provider.

b) *Encryption*

In the realm of data protection, developers often view encryption as a kind of a silver bullet, but in reality, it's just a tool—albeit a powerful one—to help achieve data protection properties. Although *full-disk encryption* (FDE) and computing on encrypted data have recently gained attention, these techniques have fallen short of answering all of the security and maintenance challenges mentioned earlier.

FDE encrypts entire physical disks with a symmetric key, often in disk firmware, for simplicity and speed. Although FDE is effective in protecting private data in certain scenarios such as stolen laptops and backup tapes, the concern is that it can't fulfill data protection goals in the cloud, where physical theft isn't the main threat.

At the other end of the spectrum, Craig Gentry recently proposed the first realization of *fully homomorphic encryption* (FHE), to which offers the promise of general computation on cipher texts. Basically, any function in plaintext can be transformed into an equivalent function in ciphertext: the server does the real work, but it doesn't know the data it's computing. Naturally, this property gives strong privacy guarantees when computing on private data, but the question of its practicality for general cloud applications still remains.

IV. FDE VERSUS FHE

A comparison of FDE and FHE in the cloud computing setting reveals how these encryption techniques fall short of addressing the aforementioned security and maintenance challenges simultaneously.

Key management and trust.

With FDE, the keys reside with the cloud platform, generally on or close to the physical drive: the cloud application user isn't involved in key management. While user data is encrypted on the physical disk, it is always accessible in the clear to any layer above it. Consequently, FDE doesn't prevent online attacks from leaking the data to an unauthorized party, which is far more common in the cloud setting than physical attacks. With FHE, untrusted applications can't easily learn or leak data. Users typically own and manage FHE encryption keys, while applications compute on encrypted forms of user data without actually "seeing" the data. This raises questions about how users can store their keys securely and reliably, especially in the presence of sharing. After all, the point of the cloud is to avoid maintaining local state.

Sharing.

Collaboration is often cited as a "killer feature" for cloud applications. Fine-grained access control is necessary to let a data owner selectively share one or more data objects with other users.

With FDE, users must fully trust the cloud provider to enforce correct access control because the key granularity (the whole disk) doesn't line up with access control granularity (a single data unit).

With FHE, because the user—or a third-party cloud provider employed by the user—manages the encryption keys, the best way of providing access control isn't clear yet. To offer fine-grained encryption-based access control, we might need to define key management on a per data object granularity basis or over collections of data objects. However, to support homomorphic operations across multiple encrypted objects, those objects must still be encrypted under the same public key.

Aggregation. Many cloud applications require performing data mining over multiple users' data for tasks such as spam filtering or computing aggregate statistics. Because users fully trust the cloud provider, performing such data aggregation is relatively easy with FDE. Current FHE techniques don't readily allow computing on multiple users' data encrypted under different keys. Therefore, it isn't clear yet how to support such data aggregation applications with FHE; similarly, offline aggregation across users' data isn't possible. One solution might be to escrow keys to the cloud provider, but that would eliminate many of FHE's benefits, making its cost harder to justify.

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

Performance. According to a recent survey, 49 percent of users abandon a site or switch to a competitor after experiencing performance issues. And the need for speed is only increasing: in 2000, a typical user was willing to wait 8 seconds for a webpage to load before navigating away; by 2009, that number dropped to 3 seconds. When FDE is implemented in disk firmware, its symmetric encryption can run at the disk's full bandwidth, effectively avoiding a slowdown. Although researchers have made significant advances in improving FHE's performance since Gentry's original proposal, it has a long way to go before becoming efficient enough to deploy at

scale. In Gentry's estimation, implementing something like a Google search with FHE would require roughly 1 trillion times more computation than the one without FHE.

Ease of development. Because FDE is hidden behind an abstraction of the physical disk, it typically has no impact on application development. In theory, FHE could also be relatively automatic: it works on an abstraction of the program as a circuit and transforms that circuit. In practice, however, performing this translation for arbitrary programs—especially when marshaling data—could be quite complex. At a minimum, programming tools would need to evolve dramatically. FHE doesn't allow developers to input data-driven judgments into the development cycle. Specifically, application developers can't look at the data, making debugging, A/B testing, and application improvements more difficult.

Maintenance. Bugs are inevitable. However, availability is a primary cloud goal, so the need to debug quickly is a top priority. Systems often fail for some unforeseen reason, requiring someone to step in and manually take action. Determining the nature of the problem might require detecting unusual activity or understanding exactly what went wrong, which isn't easy with FHE. If the application writer can't inspect application state meaningfully, debugging could be a real challenge.

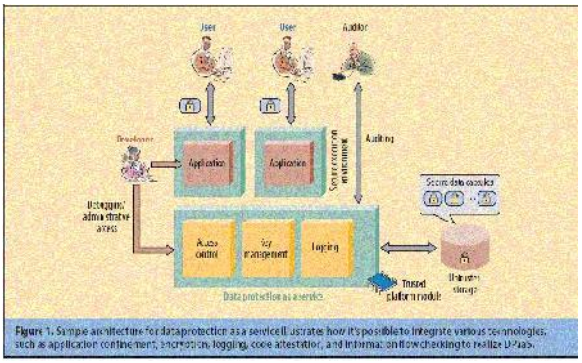

Fig 1. Project Architecture Diagram

Protection the Storage by using Diffie–Hellman key exchanging cryptographic keys in between user-to-user and user-to-Application.

V. DIFFIE–HELLMAN KEY EXCHANGE

Diffie–Hellman key exchange (D-H) is a specific method of exchanging cryptographic keys. It is one of the earliest practical examples of key exchange implemented within the field of cryptography. The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.

The scheme was first published by Whitfield Diffie and Martin Hellman in 1976, although it had been separately invented a few years earlier within GCHQ, the British signals intelligence agency, by Malcolm J. Williamson but was kept classified. In 2002, Hellman suggested the algorithm be called Diffie–Hellman–Merkle key exchange in recognition of Ralph Merkle's contribution to the invention of public-key cryptography (Hellman, 2002).

Although Diffie–Hellman key agreement itself is an *anonymous* (non-*authenticated*) key-agreement protocol, it provides the basis for a variety of authenticated protocols, and is used to provide perfect forward secrecy in Transport Layer Security's ephemeral modes (referred to as EDH or DHE depending on the cipher suite).

The method was followed shortly afterwards by RSA, an implementation of public key cryptography using asymmetric algorithms.

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

Public Key

It is also possible to use Diffie–Hellman as part of a public key infrastructure.

Alice's public key is simply **(g^a mod p ,g,p)**. To send her a message Bob chooses a random *b*, and then sends Alice **g^b mod p** (un-encrypted) together with the message encrypted with symmetric key **$(g^a)^b$** . Only Alice can decrypt the message because only she has *a*. A preshared public key also prevents man-in-the-middle attacks.

Diffie-Hellman

The Diffie-Hellman key agreement protocol (also called exponential key agreement) was developed by Diffie and Hellman [DH76] in 1976 and published in the ground-breaking paper "New Directions in Cryptography." The protocol allows two users to exchange a secret key over an insecure medium without any prior secrets.

The protocol has two system parameters *p* and *g*. They are both public and may be used by all the users in a system. Parameter *p* is a prime number and parameter *g* (usually called a generator) is an integer less than *p*, with the following property: for every number *n* between 1 and *p*-1 inclusive, there is a power *k* of *g* such that $n = g^k \bmod p$.

Suppose Alice and Bob want to agree on a shared secret key using the Diffie-Hellman key agreement protocol. They proceed as follows: First, Alice generates a random private value *a* and Bob generates a random private value *b*. Both *a* and *b* are drawn from the set of integers. Then they derive their public values using parameters *p* and *g* and their private values. Alice's public value is $g^a \bmod p$ and Bob's public value is $g^b \bmod p$. They then exchange their public values. Finally, Alice computes $g^{ab} = (g^b)^a \bmod p$, and Bob computes $g^{ba} = (g^a)^b \bmod p$. Since $g^{ab} = g^{ba} = k$, Alice and Bob now have a shared secret key*k*.

The protocol depends on the discrete logarithm problem for its security. It assumes that it is computationally infeasible to calculate the shared secret key$k = g^{ab} \bmod p$ given the two public values $g^a \bmod p$ and $g^b \bmod p$ when the prime *p* is sufficiently large. Maurer has shown that breaking the Diffie-Hellman protocol is equivalent to computing discrete logarithms under certain assumptions.

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.

To implement Diffie-Hellman, the two end users Alice and Bob, while communicating over a channel they know to be private, mutually agree on positive whole numbers *p* and *q*, such that *p* is a prime number and *q* is a generator of *p*. The generator *q* is a number that, when raised to positive whole-number powers less than *p*, never produces the same result for any two such whole numbers. The value of *p* may be large but the value of *q* is usually small. Once Alice and Bob have agreed on *p* and *q* in private, they choose positive whole-number personal keys *a* and *b*, both less than the prime-number modulus *p*. Neither user divulges their personal key to anyone; ideally they memorize these numbers and do not write them down or store them anywhere. Next, Alice and Bob compute public keys *a** and *b** based on their personal keys according to the formulas

$a^* = q^a \bmod p$

and

$b^* = q^b \bmod p$

The two users can share their public keys *a** and *b** over a communications medium assumed to be insecure, such as the Internet or a corporate wide area network (WAN). From these public keys, a number *x* can be generated by either user on the basis of their own personal keys. Alice computes *x* using the formula

$x = (b^*)^a \bmod p$

Bob computes *x* using the formula

$x = (a^*)^b \bmod p$

The value of *x* turns out to be the same according to either of the above two formulas. However, the personal keys *a* and *b*, which are critical in the calculation of *x*, have not been transmitted over a public medium. Because it is a large and apparently random number, a potential hacker has almost no chance of correctly guessing *x*, even with the help of a powerful computer to conduct millions of trials. The two users can therefore, in theory, communicate privately over a public medium with an encryption method of their choice using the decryption key *x*.

The most serious limitation of Diffie-Hellman in its basic or "pure" form is the lack of authentication. Communications using Diffie-Hellman all by itself are vulnerable to man in the middle attacks. Ideally, Diffie-Hellman should be used in conjunction with a recognized authentication method such as digital signatures to verify the identities of the users over the public communications

ISSN-2320-7884 (ONLINE)
ISSN-2321-0257 (PRINT)

medium. Diffie-Hellman is well suited for use in data communication but is less often used for data stored or archived over long periods of time.

VI. CONCLUSIONS

In this paper, a dc-coupled HPS has been studied with the three kinds of energy sources: 1) a WG as a renewable energy generation system; 2) SCs as a fast-dynamic energy storage system; and 3) FCs with ELs and hydrogen tank as a long term energy storage system. The structure of the control system is divided into three levels: 1) SCU; 2) ACU; and 3) PCU. Two power-balancing strategies have been presented and compared for the PCU: the grid-following strategy and the source following strategy. For both of them, the dc-bus voltage and the grid power can be well regulated. The experimental tests have shown that the source-following strategy has better performance on the grid power regulation than the grid-following strategy.

REFERENCES

[1] P. Maniatis, D. Akhawe, K. Fall, E. Shi, S. McCamant, and D. Song. Do You Know Where Your Data Are? Secure Data Capsules for Deployable Data Protection. In HotOS, 2011.

[2] Dawn Song, Elaine Shi, and Ian Fischer, University of California, Berkeley Umesh Shankar, Google "Cloud Data Protection for the Masses" Published by the IEEE Computer Society.

[3] S. McCamant and M. D. Ernst. Quantitative information flow as network flow capacity. In PLDI, pages 193–205, 2008.

[4] M. S. Miller. Towards a Unified Approach to Access Control and Concurrency Control. PhD thesis, Johns Hopkins University, Baltimore, Maryland, USA, May 2006.

[5] C. Dwork. The differential privacy frontier. In TCC, 2009.

[6] C. Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In STOC, pages 169–178, 2009.

[7] A. Greenberg. IBM's Blindfolded Calculator. Forbes, June 2009. Appeared in the July 13, 2009 issue of Forbes magazine.

[8] A. Sabelfeld and A. C. Myers. Language-Based Information-Flow Security. IEEE Journal on Selected Areas in Communications, 21(1):5–19, 2003

[9] L. Whitney. Microsoft Urges Laws to Boost Trust in the Cloud. http://news.cnet.com/ 8301-1009 3-10437844-83.html.

SUNIL KUMAR MANGI received his B.E Degree in **Electronics and Instrumentation** From Sir C .R .Reddy College of Engineering, Eluru, AndhraPradesh, in 2006, the M.TECH degree in CSE from Eluru College of Engineering, Eluru, in 2011-13. At present he is engaged in project titled "**Enhanced Data Protection in Cloud through Encryption Algorithms**".

Prof. G.Guru Kesava Dasu received M. Tech (I.T) Degree in 2003 and received M.E (C.S.E) Degree from ANNA University in 2008. He is having12 years of Teaching Experience in various Engineering Colleges. At present he is working as Professor & HOD, Department of CSE, Eluru College of Engineering & Technology, Eluru. He is a certified professional of IBM Rational Rose from Rational University.