International Journal of Applied Engineering Research ISSN 0973-4562 Volume 14, Number 13 (2019) pp. 3003-3016

Survey on Unsolved Informational Security Research Issues and Purported Solutions for Data Security in Public Cloud

P.Karthik[1], Dr. P.Shanthi Bala[2]

[1]Research Scholar, [2]Assistant Professor,
Department of Computer Science, School of Engineering and Technology,
Pondicherry University, Puducherry, India.

Abstract

Information security is the prime concern of the modern computing environment, as every user in the universe has to depend on the internet for their day to day operations. The technological advancement in the domain of Information Technology leverages user community to make of this technology to its fullest potential without posing any limitations on them. Therefore, this scenario inevitably increases the storage and computational requirements of the end user. This is the actual driving force that makes the industry to shift towards the cloud paradigmatic solutions. But the biggest challenge in cloud computing solutions is the loss of physical control over the user data which raises concerns about security and privacy of the stored data. This paper addresses data security concerns over four major domains namely Data Transmission, Virtualization, Distributed Storage and Remote data integrity. This paper also tried to provide purported solutions for data transmission, virtualization and integrity issues in distributed storage.

Keywords: Security Issues of Cloud, Unsolved research issues in public Cloud, Data Security Issues, Issues on Storage Distribution, Vulnerabilities of Hypervisor

1. INTRODUCTION

Cloud computing is the new paradigm that has completely changed the style of today's computing and its environment. Every business holders ranging from small companies to very big corporations have already started shifting their data to this new technology [1][2]. Some of the real benefits of cloud include any time anywhere access, increased data availability, robustness, quick application development, scalability, dynamic provisioning, and reduced operational cost. This new paradigm shifts much of the organizational burdens into shoulders of the cloud service provider and hence enables the application developer and cloud users free from worrying about issues related to the performance of the applications, security, and other infrastructure facilities needed for quick application development. Though cloud offers low-cost solutions to the best possible ways, it also introduces new problems in the domain of security that haven't brought up to the notice before. This technology necessitates the user/organization to move their data to the cloud. For an organization the volume of data is huge and has to be transported through an unreliable public medium called the internet. It opens up new possibilities for the hackers to attack the organizational data during the transit. Service level agreements emphasize security as the core issue but most of the time it covers only the stored data in the cloud and not the data in transit. In addition to this, the possibility of data outages and leakages are high in cloud environment due to design flaws and poor security policies [3][4]. This paper addresses various unsolved informational security research issues of cloud and provides purported solutions for them.

The paper is organized as follows; Part 2 deals with data communication issues, part 3 explains virtualization vulnerabilities, part 4 elaborates issues of data distribution, part 5 explicates remote data integrity issues and part 6 deals with a conclusion and future works.

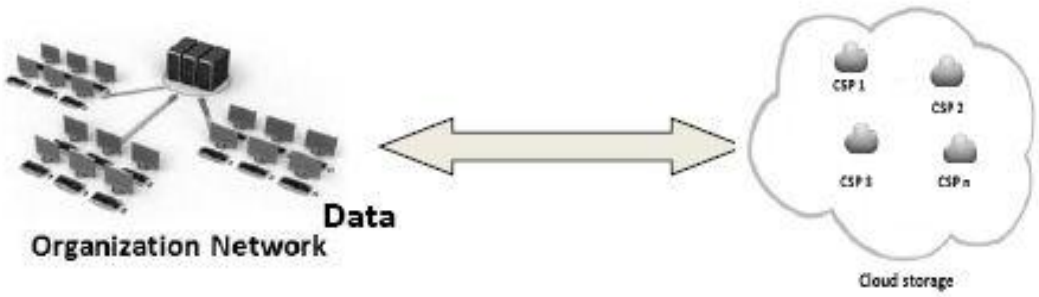

Fig 1: Typical data flow between organization and Cloud

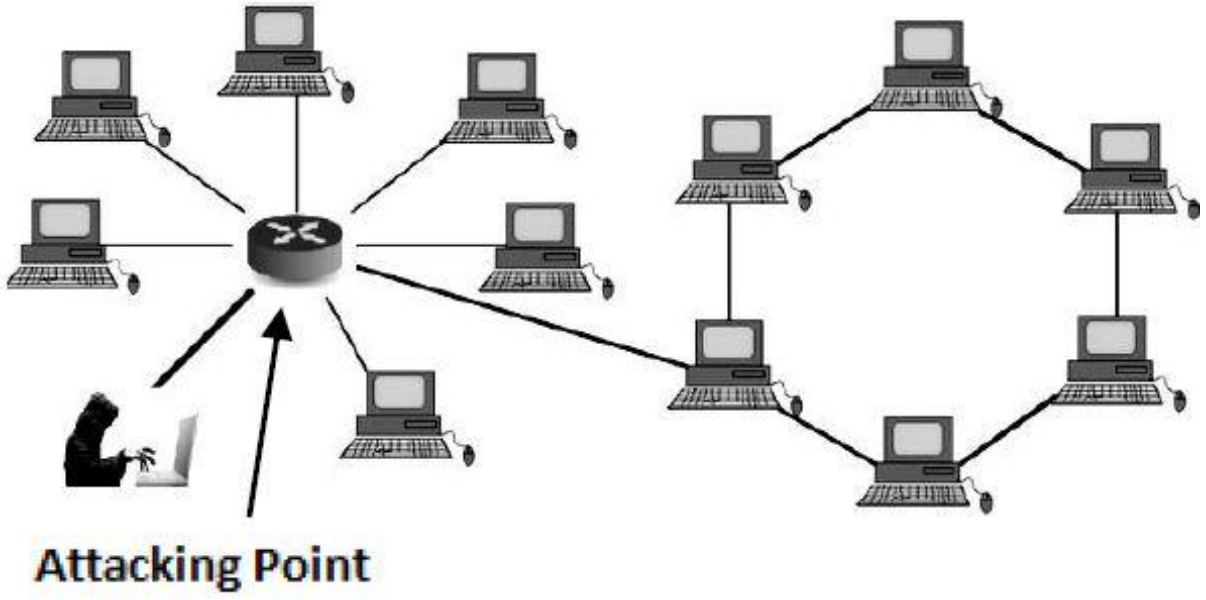

Fig 2: Unauthorized access at the gateway

2. DATA COMMUNICATION ISSUES

Data communication here refers to the movement of data among different entities involved in the cloud paradigm. The entities are user, organization, cloud service provider and unreliable public medium i.e. the internet. The data movement usually happens in the following situations.

- User/organization when import data to a cloud infrastructure
- User/organization when import data from one service provider to another service provider.
- Between different cloud service providers on sharing the resources among the service providers.

2.1 Data Access

To ensure quick data transfer the connection should be fair enough to support constant high data transfer rate. But it is impossible to obtain it from the internet as the behavior of the internet is unpredictable and its speed depends on the current traffic of the network. Therefore, one cannot expect constant performance from the network and thus from the cloud. The worst case happens when the user is doing some critical operations using the cloud resources or the entire network is fully down. In such circumferences, the user might not continue his work, or he would be completely prevented from accessing cloud services due to access failure. Hence, the reliability of the cloud service cannot be assured with a guarantee.

2.2 Data Integrity and Privacy

The term integrity refers to the consistency of the data under consideration. The data integrity aims to prevent any data modification by an unauthorized person. Data privacy deals with protecting the intelligence of the data only with intended users. It is really a challenging task to ensure these properties to the data when transmitting through the unreliable medium. It's a known fact that the Internet works based on packet switching technology, which allows sharing of the medium among different users to ascertain maximum resource utilization. Therefore any person who has access with an internet can able to see or access the data without the knowledge of the sender and receiver. Particularly, if the intruder is from the same network then collateral damage will happen to the integrity of the data as he is able to access the entire data using techniques like MAC flooding. This is the clear cut violation of integrity.

2.3 Conventional Security Procedures: An overview

Security is the major concern of data transmission over the internet as well as storage in the cloud [5][6][7][8]. Encryption algorithms are deployed for ensuring security and privacy for the data. The problem with this approach is; it is time-consuming and requires too much computational power to convert it into chipper text.

Let S be the Security of the system under consideration, C be the complexity of the cryptographic algorithm and T be the execution time of an algorithm and V be the volume of data. Therefore the security of the system can be expressed as follows:

$$S \propto C \quad \text{……..(1)}$$

$$C \propto T \quad \text{……. (2)}$$

$$V \propto T \quad \text{…… (3)}$$

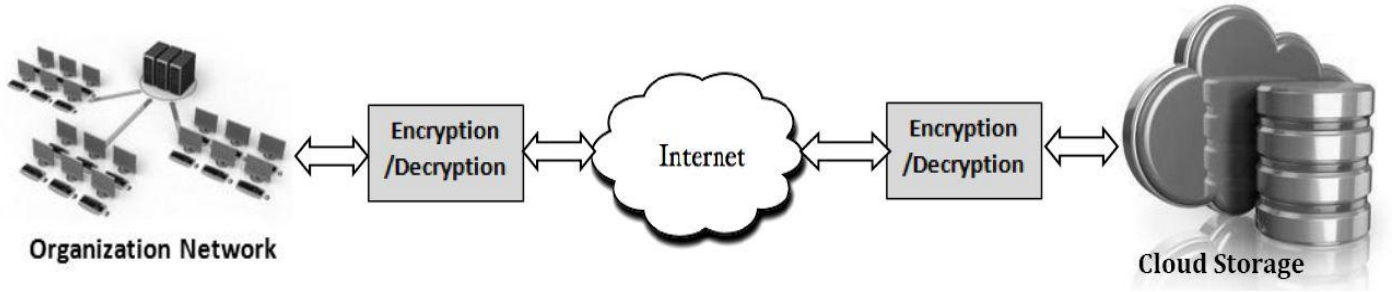

Fig 3: Data Storage and retrieval from cloud storage

Hence, to improve the security more complex algorithms need to be deployed in the process of encryption and decryption on both ends. But involving complex algorithm necessitates high configuration machines with huge storage requirements. Meeting the aforesaid criteria may be impossible for the client which is generally having a low configured machine. Further, encryption and decryption algorithms itself not adequate for the security of the data to be transmitted as no algorithm in this world claims to be unbreakable.

Https is an option to provide security over transmitted data. It uses TLS/SSL for performing encryption and decryption. But this approach also proved to be ineffective in providing data security. The main reason for the failure is TLS/SSL still uses the insecure RC4 algorithm for nearly half of its traffic and it could not use complex algorithms due to network performance considerations. Therefore, to provide a comprehensive solution for data security lightweight algorithms to be developed without compromise in security.

2.4 Security Features of Cipher Algorithms

The conventional cryptosystems use two techniques to perform encryption and decryption.

- Symmetric Cipher
- Asymmetric Cipher

Symmetric cipher uses a unique key to perform encryption and decryption at both ends. It produces cipher text using XOR operation along with bit manipulation techniques (shifting and moving). The reverse of encryption will produce an authentic message at the receiving end. These ciphers are relatively barely secure but faster than asymmetric algorithms.

Asymmetric ciphers use two keys (public and private keys) to perform encryption and decryption. Encryption uses public key and decryption uses a private key. The confidentiality of the private key is preserved by not sharing it with anyone. Asymmetric cipher involves huge computation to perform encryption and decryption and is resource intensive. These ciphers are secure but slower than symmetric algorithms. Therefore, the trade-off should be made in choosing the algorithm.

2.5 Cipher Text Attacks

The rigorous analysis of all chipper-text attack [9] reveals the following facts.

- Encryption algorithm uses keys of discrete length to perform encryption.
- The similar key will produce similar cipher-text.
- The encryption algorithm transmits the cipher-text data directly to the recipient via an insecure public medium.
- Neither the network nor the encryption algorithm prevents the intruder to gain access to the cipher-text.

Therefore, it provides ample opportunities for the attackers to sneak/snoop the data without the knowledge of the sender and receiver [10]. As the packets travel along different paths to reach the destination, it might be impossible for the intruder to receive all the packets on transit. But substantial possibilities are available for an intruder to have access to few of them. The worst case happens when the intruder attaches himself with the same network from which the packet originates. The encryption algorithm can prevent the intruder from extracting the intelligence of the data only for a limited period (up to the time taken for an intruder program to break the key using known cipher-text attacks). To launch a successful attack, the user would collect some useful information like name and type of the encryption algorithm, name of the client and his personal information such as date of birth, etc. When sufficient information is available to an intruder then he might launch a successful attack.

For example, to successfully launch a brute-force attack an intruder would possess the following information.

- Name of the Encryption Algorithm.
- Length of the key.
- Cipher-text, plaintext pair

All the encryption algorithms are aimed to increase the break-up time so that the information might be unavailable in the hands of the wrongdoer in the near future. Hence to solve the problem related to data security the following points to be considered.

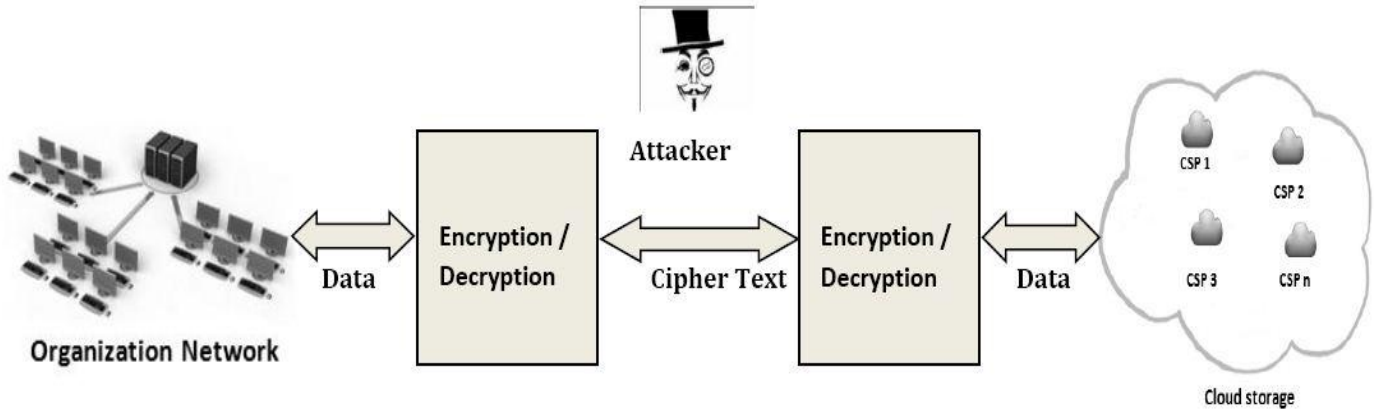

Fig 4: Encrypted Data Transfer for security and privacy over transmitted data

- **Preventing the Information not to available as a whole to an unauthorized person.**

This could be performed by splitting the message and transmit the split information via divergent paths (using two different transmission techniques namely wired and wireless). The drastic improvement in the field of wireless communication and the realization of 3G and 4G technologies provides a reliable platform to implement this.

- **Make the footprints of the conventional algorithms difficult to trace.**

Given the same message and key conventional cipher algorithms produce similar cipher-text. This enables an intruder to set conditions for an attack. If an intruder is capable to discover the authentic text for some portion of the cipher-text, then he could be able to break other portions of the message irrespective of whatever the length of the key. Conversely, if an algorithm is designed to produce different footprints for the given message and the key then setting conditions will be computationally infeasible for an intruder to break.

By incorporating the aforementioned ideas, we propose the following algorithm that enhances the security of the encrypted data over the mistrusted communication medium.

In addition to this, it provides an opportunity for the sender to go with lightweight encryption techniques to minimize the computational overhead from the client end. To implement this, it is presumed that nodes involved in the transmission have to support wired and wireless transmission. Let **M** represents the message to be transmitted.

Procedure Split_Security (M)

Begin

Step 1: *Split the message* M *into* m *discrete blocks of size* n.

$M = \{ B \mid B = \{b_1, b_2, b_3, .., b_m\}\}$

$\forall$ b_i B, *calculate*

$sp_1 = \{t_{11}, t_{21}, t_{31}, \ldots t_{n1}\}$

$sp_2 = \{t_{12}, t_{22}, t_{32}, \ldots t_{n2}\}$

Such that $b_i = \{sp_1 + sp_2 \neq sp_1 \quad sp_2\}$ *and*

$B = \{ \forall \mid \in y_i \ \exists y, \in _i \quad Z \mid y_i = P(y_i)$ *and* $P(b_i) = P(y_i)$

Step 2: $\forall \quad y_i \in$ *Y, split* y_i *into two unequal parts.*

$F(y_i) = \{(y_{i1}, y_{i2}) \mid y_{i1}, y_{i2} \in Z$ *and* $y_{i1} \neq y_{i2}$

and $y_{i1} + y_{i2} = y_i\}$

Step 3: $\forall\forall \ b_i \in B$, *calculate*

$esp_1 = EA(sp_1)$, $esp_2 = EA(sp_2)$

also, $\forall \ y_i \in Y$ *calculate* $ey_i = \{ey_{i1}, ey_{i2}\}$

$ey_{i1} = EA(y_{i1})$, $ey_{i2} = EA(y_{i2})$

Where, EA is a light weight encryption algorithm.

Step 4: *Send* esp_1 *and* esp_2 *after shuffling via different paths namely* P_1 *and* P_2 *such that* $P1 \neq P_2$

End;

Note: *The paths P1 and P2 are chosen such that no two nodes in the path should be repeated again except source and destination. (ie)* $P1 \cap P2 = \{source, destination\}$.

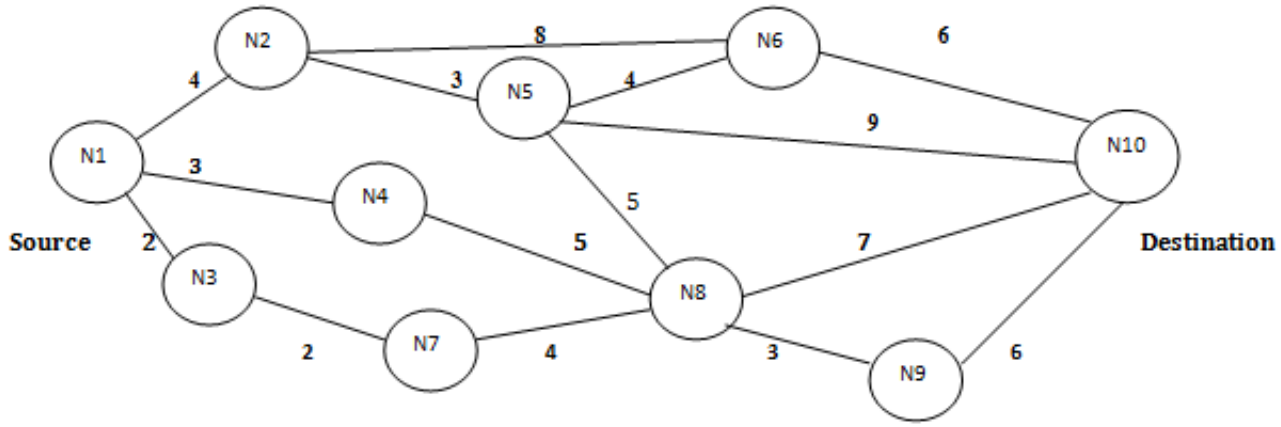

Fig 5: Finding alternate path for the Data Transfer of Split Security Algorithm

In the receiving end, the same encryption algorithm is used to decrypt the values from esp_1 and esp_2. These values are then subsequently used to find the original value **bi** in the receiving end. To ensure the integrity of the transmitted data unique tag value (ie)***Y*** is generated for the entire message ***M***. The tag is usually generated using digest function and is sent to the recipient. In the receiving end, the value is calculated back and is verified with the received value. If the tags are equal, then integrity is achieved. In this approach, the hacker has no way to identify the intelligence of the data.

2.6 Path Selection for Split Security Algorithm

To transmit the split data to the destination, it is inevitable to discover two divergent paths P1 and P2 such that P1 ∩ P2 = {Ns, Nd} where Ns and Nd refer the source and destination nodes respectively. The fundamental idea behind the data split is to harden the possibility of the entire/part of the cipher-text packets available in the hands of an intruder. This would not merely improve the security of data but also prevents all forms of cipher-text attack on the transmitted data. Let N be the number of nodes and P represents the set of paths between the source and destination. N={N1,N2,N3,N4,N5,N6,N7,N8, N9,N10}. The following table represents the various paths between the source and the destination.

P={P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13,P14,P15}.

From the path set P, the first path has to be chosen such that cost for the path is as minimal as possible. Path P6 and P11 involve cost least from the table. Therefore they could be considered for data transfer. Though both paths involve the similar cost factor, P6 is given precedence as the number of hops involved is 2.

Table 1: Global Hash Table Entries

Path	Nodes on the Path	Cost
P1	{ N1,N2,N6,N10}	18
P2	{N1,N2,N5,N6,N10}	17
P3	{ N1,N2,N5,N10}	16
P4	{ N1,N2,N5,N8,N10}	19
P5	{ N1,N2,N5,N8,N9,N10}	21
P6	{N1,N4,N8,N10}	15
P7	{N1,N4,N8,N9,N10}	17
P8	{N1,N4,N8,N5,N10}	22
P9	{N1,N4,N8,N5,N6,N10}	23
P10	{N1,N4,N8,N5,N2,N6,N10}	30
P11	{N1,N3,N7,N8,N10}	15
P12	{N1,N3,N7,N8,N9,N10}	17
P13	{N1,N3,N7,N8,N5,N10}	22
P14	{N1,N3,N7,N8,N5,N6,N10}	23
P15	{N1,N3,N7,N8,N5,N2,N6,N10}	30

The further step is construct a subset of the path set P_{sub} such that P_{sub} should not contain the intermediate nodes present on the selected path(i.e) P6.

Table 2: Global Hash Table Entries for P_{sub}

Path	Nodes on the Path	Cost
P1	{ N1,N2,N6,N10}	18
P2	{N1,N2,N5,N6,N10}	17
P3	{ N1,N2,N5,N10}	16

N_{sub} = {N1,N2,N3,N5,N6,N7,N9,N10}.

P_{sub} ={P1,P2,P3}.

P3 is chosen as a second path as it involves the least cost among the present paths. Ultimately, the Split information is transmitted via two paths namely P3 and P6. At any point of time, the intruder could be barely accomplished to receive the packet containing partial information of the cipher-text. Therefore, launching a cipher-text attack becomes computationally infeasible. The barely possible way to perform any cipher-text attack on the transmitted data is to attach self in any of the intermediate nodes in both paths at the same time. As the geographical location is exotic and the network latency further complicates the issue, then performing all forms of cipher-text attack would be practically infeasible.

2.7 Challenges and Feasibility Analysis of Split Security Algorithm

The proposed approach solves much of the integrity and security issues of data on transit, but the use of two different transmission modes for the data transfer introduces additional overhead during the reconstruction of the authentic message. To effectively reconstruct the authentic message at the receiving end, the message should be received on either path in time. As the behavior of the wireless transmission is often unpredictable (Its speed is subject to the air traffic and bandwidth factors of the channel.), one cannot expect constant throughput out of the system. Therefore, this method might be unsuitable for the transmission of large data chunk where constant throughput remains the crucial factor. But the realization of 3G and 4G technologies opens up possibilities for the implementation of the proposed design.

3. VIRTUALIZATION VULNERABILITIES

3.1 Introduction

The term virtualization was initially coined by IBM in 1960 for the first time. The cost of the hardware devices was exceptionally significant at that time and hence to perform baffling computations powerful machines such as mainframe were needed to be deployed for such tasks. The investment cost factor played a crucial role and therefore the computing industries were in a need to make use of these precious resources in an effective way as the cost factor prevented them further to purchase experimental machines in large numbers. This actually paved the way for the invention of a modern technology called virtualization. Virtualization is an abstraction technique which basically relies on resource utilization and sharing techniques.

During 1980, the hardware industry was revolutionized to its peak and as a result of this, the cost of the hardware resources started decreasing. Therefore the motivation of virtualization gets stammered and no further research had been done during the prior period. The rebirth of virtualization started afresh with the emergence of client-server computing during the 1990s. The alternative paradigm has transformed the way the computation had been performed in the past. Security, Reliability, performance, ease of administration with fine-grained user control, increased complexity with easy user interface components and less power consumption and heat dissipation have been given much more importance in the experimental paradigm. Virtualization addresses all the aforementioned problems by abstracting all the underlying resources and presents them to virtual machines as if they were obtained their own set of resources.

3.2 Virtualization: The Challenges

The art of virtualization offers several benefits. Reduced capital expenditure, improved resource utilization [11], less power consumption and reduced maintenance are worth to mention a few. Besides, it introduces some serious challenges in the deployment of virtual machine isolation [12], obtaining improved overall system performance [13] and providing security to VM's Data [14]. Each virtual machine is provided an illusion that as if they were provided with own set of resources. So the virtual machines tend to consume the resources without any limitations imposed on them. If one or more virtual machines utilize the resources to its maximum need then it may impact the performance of other virtual machines running on the same set of hardware resources. Maintaining the overall performance of the system is another challenge in a virtualized environment. Para-virtualization may slightly improve the performance but it requires a modified operating system. Full virtualization doesn't require this modification but the bulky size of privileged instruction executed by the virtual machine requires translation by virtual machine monitor or hypervisor thus subsequently affect the overall performance of the system. The data security of the cloud is an another concern. The data outages[1] confirms the fact that the security of the cloud is breakable[2].

3.3 Virtualization: Security Concerns

Security remains a critical concern in a virtualization environment. Virtualization allows resource sharing among the virtual machines. This introduces new security loopholes in the execution environment which subsequently raises a concern about data security.

1. http://www.databreachtoday.com/news#
2. http://www.cio.com/article/2954194/russian-hackers-use-twitter-to-mask-sneaky-data-theft.html

Though virtual machine isolation prevents one virtual machine to interfere with other virtual machines, memory overcommitment, however, allows the pages to be shared between virtual machines. This section addresses the security aspects of virtualization with respect to the deployment of virtualization architecture. The deployment of virtualization can be served in two ways namely Type 1 or bare metal architecture and Type 2 or hosted architecture.

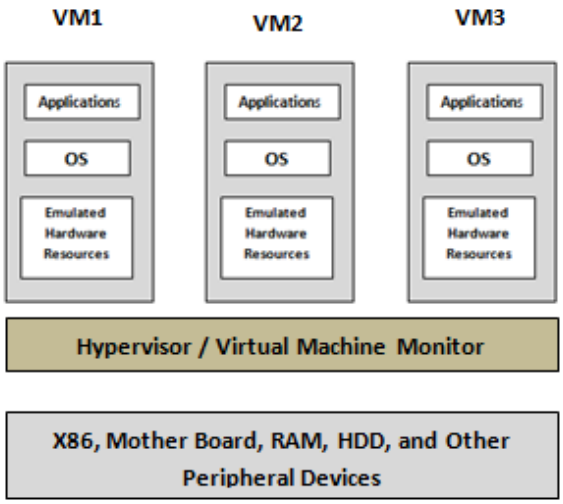

Fig 6: Type 1 or Bare Metal Architecture

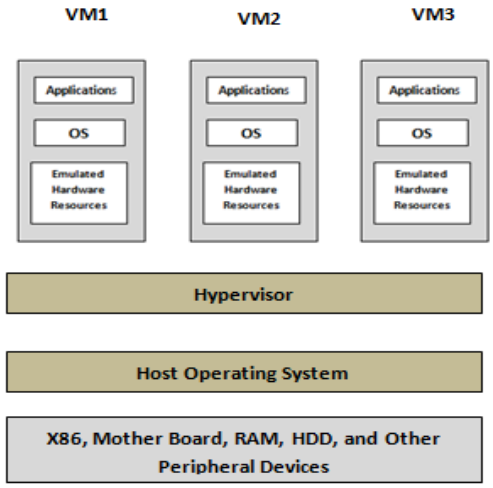

Fig. 7: Type 2 or Hosted Architecture

The type-1 architecture contains three layers. The top layer contains virtual machines with heterogeneous operating systems and application programs. The middle layer is a thin layer and responsible layer for the allocation of resources to the virtual machines. This Layer runs the management OS (Management OS can also be called as Virtual Machine Monitor.) and that is responsible for controlling the operations of other virtual machines. The problem with this design is the size of the management OS/Hypervisor and is comparatively very high due to its increased functionalities. Because of the large code base, the interfaces for the integration of functional components in the management operating system can be exploited. Since all the virtual machines are controlled/managed by the management OS, it is very easy to access any confidential data stored in the virtual machines if one can able to compromise the management OS. Even the encryption techniques fail to provide security for the integrity of the stored data as the keys can be easily obtained by accessing the pages pertaining to the target virtual machine.

The type-2 architecture contains one additional layer called Host OS Layer. This layer actually controls the underlying hardware resources. In this design, the virtualization layer appears to be an application program for the host OS. The significant part of the design is the hypervisor acts as an intermediary between the guest operating system and host operating system. This enables virtual machine portability besides introducing a new problem that the state of the guest or host machine is not known to the hypervisor. Therefore, it could not have a control over both guest and host machine and it is only responsible for mapping of the virtual address to the machine address translation [15][16][17]. Here virtual address is the address generated by the application programs in the virtual machine. Machine address is the address of the real address of the underlying hardware resources. Physical address space represents the emulated hardware resources generated by the hypervisor. The guest operating system maps the virtual address space into physical address space. The state of the virtual machines is stored as virtual disk images in the host operating systems. These images can be easily accessed even without authentication from the host operating system like ordinary file access.

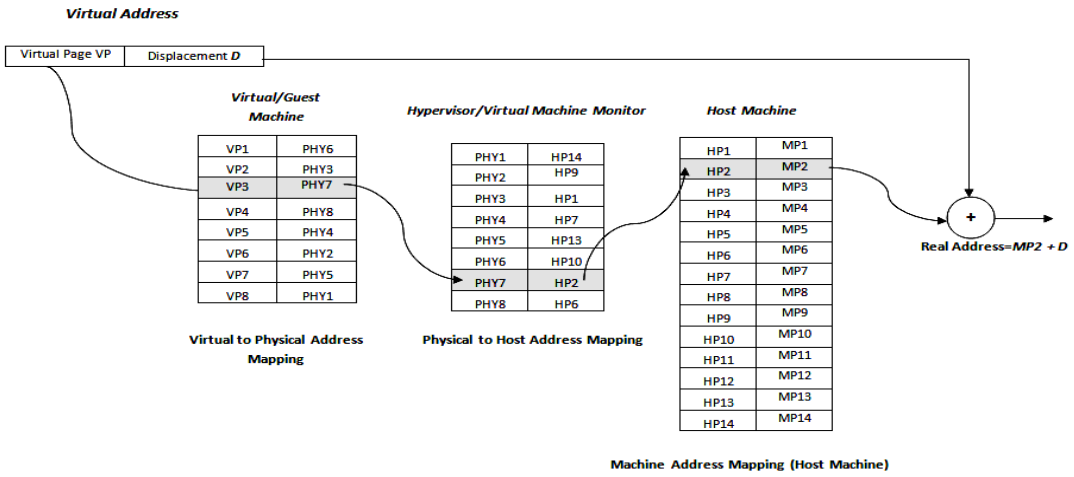

Fig 8: Address Translation Mechanism in Type II Hosted Architecture

3.4 Virtualization Memory Management

Memory is the costliest and scarce resource that needed to be efficiently managed for system performance. Typical memory management in virtualization revolves around memory overcommitment which enables the virtual machines to be provided with more memory than the available physical memory. In overcommitted memory state;

$\sum_{i=1}^{n}$ VMPMem$_i$ > Available Physical Memory

At this place, VMPMem$_i$ refers to the physical memory of the ith virtual machine.

The problem associated with memory overcommitment is resolved by employing the following techniques [18][19][20].

- Memory Sharing through Transparent Paging System(TPS)
- Memory Ballooning
- Hypervisor Swapping
- Memory Compression

The particular technique deployed for memory reclamation depends on the state of the memory. There are four memory states high, soft, hard and low which represent the amount of free host memory available in a particular instant. To be marked as high memory state, 6% of the total host memory would be free. Identically, this would happen to be 4 % for the soft state, 2 % for the hard state and 1 % for the low state.

a. Transparent Paging System (TPS)

The Transparent paging system is the popular memory reclamation technique used in the hypervisor/virtual machine monitor [21]. In ESX server the transparent paging system is the default technique employed when the ESX memory state is high. This approach uses a global hash table common for all virtual machines running under the hypervisor.

Table 3: Global Hash Table Entries

Hash Value	Pointer to the Page in Host Memory
1256788	12288
4523145	36864
.	.
.	.
.	.
3241567	40960

When a new page is brought in to host memory then it calculates the hash value of the incoming page and checks for identical memory contents by comparing the hash value of the new page with global hash table entries. When a match occurs, it subsequently performs byte by byte memory comparison for an exact match. Subsequent to the exact match it subsequently modifies the page table entries such that two or more page contexts currently refer the same physical memory. Therefore, this approach prevents the unnecessary loading of redundant copies in the primary memory thereby effectively handles the memory overcommitment problem. The complete sequence has been presented in the procedure given below.

Procedure TPS (p)

Begin

Step 1: Calculate hash value of p as
HVp = Hash (p)
Step 2: Search HVp value in the global hash table.
Step 3: If (match for hash value = true)
{
Perform byte by byte content comparison of P with matched page for exact match.
If (exact match found)
{
Modify the page table entries of the new context so that two or more contexts now refer the same physical frame in the host memory.
}
}
Else
{
Update the Global hash table entries with the values Of incoming page.
}

End;

The drawback of the design is if the hypervisor/virtual machine monitor is compromised then the mapping of identical page contents could be exploited so that any virtual machine pages currently could be read or monitored with ease and comfort.

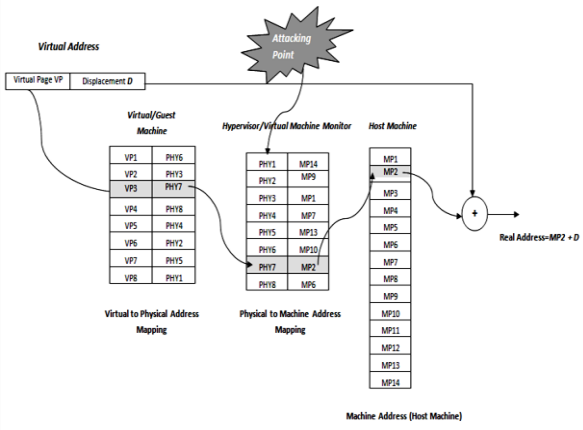

Fig 9: Hypervisor Attack on Page map Table (Type I)

Another concern about the security is the TPS considers only about identical contents of virtual machine pages and never consider about service level agreement or the privacy of the user data. Virtualization provides only the illusion to the users as if security were provided for user data by allocating

individual resources. In contrast, it shares the underlying resources with other virtual machines perhaps with other users without the knowledge/permission of the users. It is the violation of confidentiality of the stored data in the cloud because of the reason that any data to be processed should be brought into primary memory. Therefore, the virtualization layer itself has to be modified that it has to perform memory allocation in tune with service level agreement made by the user. The extra penalty may be levied on users for separate memory allocation for the user's process.

b. Memory Ballooning

Memory ballooning [20][21][22][23] is an alternate technique enabled when the memory state of the ESX server is soft. Each virtual machine is placed with a separate balloon driver which is used by the host operating system to reclaim memory. The Balloon driver does not have any external interfaces so that any application program running inside the virtual machine cannot communicate with a balloon driver. Each balloon driver is provided with a private channel through which the virtual machine communicates with the hypervisor. The balloon driver scans the virtualization layer periodically to get the target balloon size set by the host operating system. It subsequently performs memory reclamation to meet the demands of the balloon size. Allocation will be performed from the free page pool maintained by every virtual machine. If enough free pages are unavailable in the free page pool then it invokes procedures to reclaim memory until it gets the target balloon size. It is a time-consuming process but can be achieved without affecting the performance of other virtual machines. The continuous sequence of the memory balloon mechanism could be listed as follows.

- The user invokes application from the virtual machine send memory request to the guest operating system.
- The memory request will be noticed by the hypervisor/virtual machine monitor. It in turn forward the memory request to host operating system if it is hosted architecture otherwise it itself allocate memory from the free memory pool if it is bare metal architecture.
- The memory will presently be provided to the application, and the page map tables are updated in the guest machine (Virtual address-> Physical address), Hypervisor/Virtual Machine Monitor (Physical Address->Machine Address) and Host machine(Host Virtual address->Machine Address)
- When the user closes the application in the virtual machine the guest OS reclaims the memory, update necessary page map table entries in the Guest OS and finally add them to free memory pool for reuse to other applications. But this information would be unavailable neither to the hypervisor nor to the Host operating system due to virtual machine isolation.
- The balloon driver is used to reclaim such free memory from the virtual machine so that it would be made available to the host operating system.
- Upon realizing the memory pressure of the host operating system the hypervisor sets the target balloon size for every virtual machine. The virtual machine pins the free memory pages for handing over the same to the hypervisor. This process is called inflate. The inflate-mechanism would make the balloon size bigger in tune with a target balloon size.
- Now the hypervisor reclaims those memory pages from the virtual machine through a hyper call called deflate. The deflate-mechanism would reduce the balloon size smaller.
- Now the hypervisor comes knowing the free memory pages so that the information would now be updated in page map tables of the hypervisor and host machine.

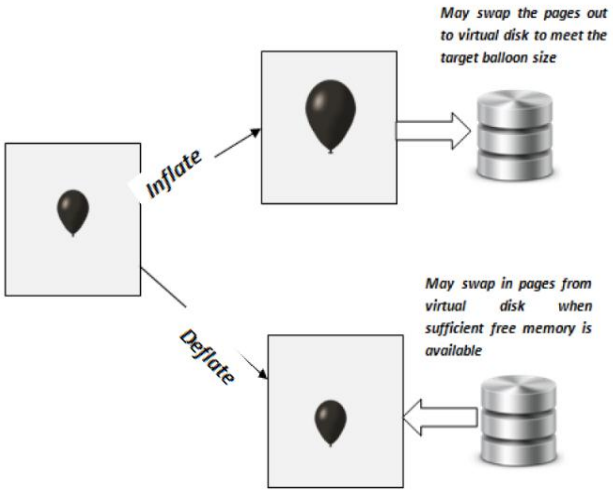

Fig 10: Memory Balloon Mechanism in ESX Server

The problem with this approach is that the virtualization layer will never know when it will get the required pages from the balloon driver. It works well when the ***target_balloon_size*** is less than the size of the ***free_page_pool***. Another drawback of this approach is that, if the Hypervisor/Virtual Machine Monitor is compromised then one can modify the ***targer_balloon_size*** variable by exploiting the private channel and subsequently launches DOS [25] attack on all the virtual machines.

Procedure Memory_Balloon()
Begin
Step 1: Read the target_balloon_size.
Step 2: if(target_balloon_ Size<=Free_Page_Pool_Size)
{ Pins the free pages and infirm the pined page numbers to virtualization layer by freeing the pages from free page pool.
}
Else
{
While(target_balloon_size > Free_Page_Pool_Size)
{
Invoke memory reclamation Algorithm to meet the demands of the target balloon size.
}
}
End;

c. Hypervisor Swapping

Hypervisor swapping is another memory reclamation technique and is enabled when the ESX memory state is hard. This technique provides a time guarantee for memory reclamation but involves substantial performance overhead in the execution of virtual machines. This design enables the hypervisor to randomly choose pages from the virtual machine and page out them into the swap area of the corresponding virtual machine.

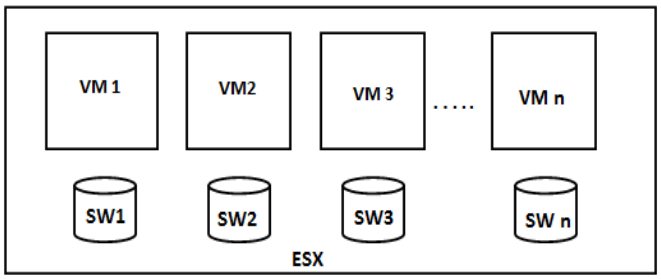

Fig 11: Hypervisor swapping

The design lags when the randomly chosen pages are happen to be the most frequently used pages or kernel pages of the guest operating system. It results to poor system performance due to frequent page fault and as the worst case it will lead to virtual machine crash. From security point of view this never should be practiced as the user will lose the entire data. Least frequently used pages may be the good choice for swapping but it will not always give good performance. Identifying the kernel and most frequently used pages are also extremely difficult due to virtual machine isolation.

d. Memory Compression

Memory compression is a technique which reclaims the memory by compressing the two or more memory pages into a particular page. The partially filled pages are the good choice of selection (Typically, this happens to comprise the ultimate page of the process.) for memory compression. This is furthermore a time-bounded process which would produce sufficient memory for the host machine but engenders significant performance overhead in the execution of the virtual machines. The problem with this approach is that, if the pages selected for compression are happened to represent the kernel or most frequently used pages, then severe performance overhead will occur. The compressed pages can't be managed by the active process directly though it is present in its physical memory. It has to be decompressed back to use again. The complexity of the compression algorithm also introduces additional overhead to CPU. Particularly the situation will get worse when there are no free page blocks available for the virtual machine to place the decompressed contents back into the primary memory for processing. This might lead to system crash and hence it never should be practiced for memory reclamation.

4. CLOUD STORAGE: DATA DISTRIBUTION ISSUES

4.1 Data Distribution in Cloud

The success of cloud paradigm revolves around the following factors; cost-effective solutions, any time anywhere access, reliability, on-demand provisioning of resources, scalability, and increased data availability. Therefore, it is imperative for the cloud system to implement distributed redundant storage in support of reliability and its associates called increased data availability and access. To achieve this into reality, the redundant storages must be geographically distributed even to withstand against natural calamities like an earthquake, Tsunami, etc. This introduces renewed concerns on security over the stored data in the public cloud with respect to confidentiality, integrity, and authenticity[24,25,26,27].

The issues on confidentiality and authenticity could be addressed through the existing techniques namely authentication and encryption [28]. Hence, this paper proposes solutions related to data inconsistency issues arise out of data replication in public cloud[29][30].

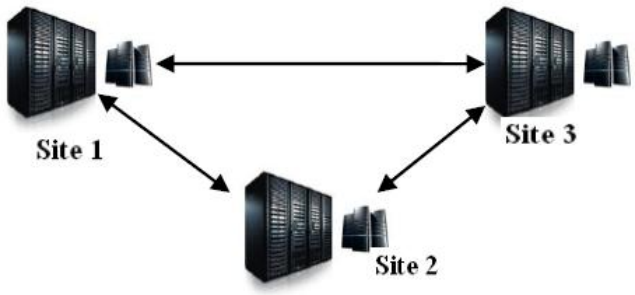

Fig 12: Data Centers managed with one or more servers

Figure 12 shows the typical storage structure of public cloud managed by one or more servers with huge storage support by the datacenters. This design may use location-based redundancy which moves the data near to its usage to minimize the response time of the server. When a request originates from a user, it would be directed to the nearby server so that a user would get a quick response from the server. The concept of anytime anywhere access permits the user to move around the globe so that it is possible for the users to perform huge data manipulation at one site and subsequently moved to another site. Particularly such incident may happen in a public cloud in larger numbers. Hence, huge data transfer is required among the data centers to maintain consistency among the replicated instances. As the geographically distributed data centers are often connected via the network, it might be impossible for the server at one site to update these modifications in other sites i.e. replica sites. The process of a complete update might typically take from few seconds to several hours. Meanwhile, if the data is being accessed from other sites the user may possibly get wrong data or encounters problems like non-availability of data.

4.2 Proposed Solution

This paper proposes a solution for data inconsistency and non-availability problem of redundant cloud storage through intelligent agents [15][16][17]. At this place, the agents are the loosely coupled autonomous servers backed up with huge storage support (i.e) data centers and they cooperatively work to perform a task.

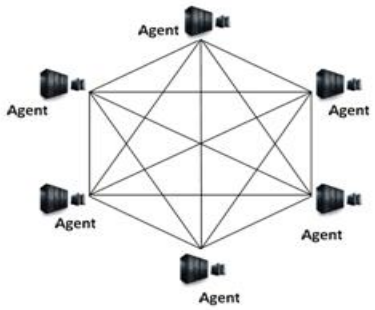

Fig 13: Cloud storage managed with Redundant Multi-Agents.

The agents are connected with one another as shown in Figure 13. Each server in the agent is assigned with a static IP address. To support reliability, the agents would store the redundant information at multiple sites. Every agent will have information about other agents. The agents are provided with well-defined interfaces through which the data can be retrieved. Whenever the user login to an agent, the server monitors the activities of the user. Any attempt to perform insertion/modification in the storage/stored data will be identified and message will be sent redundant agents for possible subsequent actions. The message comprising information such as IP address, name of the file/table to be inserted or modified, a user login details and login time. This would enable the redundant agents to know about the changes that happened at one site. Upon receipt of the message, the redundant agents now update the user's data in its storage by submitting a special request to an originating server. If immediate update is not possible then the redundant agents affix the message as tags along with user login details. The complete sequence is depicted as Figure 14 and Figure 15.

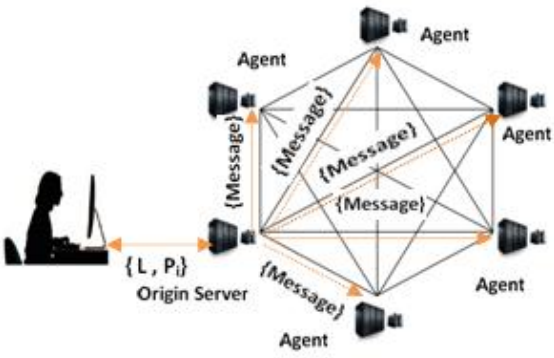

Fig 14: Message propagation for redundant servers

Let **P** be the set of agents such that **P={ p_1,p_2,p_3,…p_n}. L** be the user supplied login information and **CL** be the login information retrieved from the agent/server.

Procedure Data_Access(L ,p_i)

Begin

Read CL and Tag from p_i.

If(L=CL and Tag= Ø)then

{

Allow access to cloud data and monitor the user operation for insert, update, upload and delete.

If (modified=true)

{

Compose message as

Message= {IP, name of the file, login details, Login time}
Propagate message to redundant agents R such that R ⊂ P

}

}

If (Tag ≠ Ø)

{

Redirect request to IP

}

End;

The redundant agents periodically check the ***Tag*** value of every cloud user and makes special request to appropriate originating servers.

Procedure Update_ Redundant _Agent (L, names of the files)

Begin

Upload modified contents pertaining to log in credentials L on redundant servers;

End

After a successful update, the redundant agents promptly remove the tags from user information. As message size is comparatively small than the authentic data, propagation could be easily performed within a brief time. It is possible for a user to transport data to another site and tries to access from the new location. As mentioned earlier the request would be directed to a nearby server. Upon the receipt of the user login details. At this point, the redundant server retrieves user login details from cloud storage for authentication. If it recognizes any valid tag along with user login details then the server knows that the current information is present with the originating server. So it would redirect the request to an originating server so that the user will now get a response from the originating server.

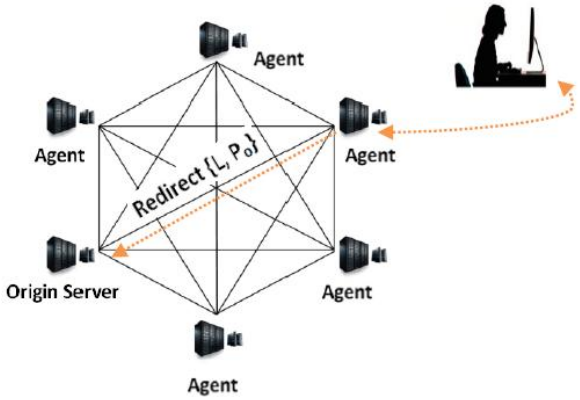

Fig 15: Request Redirection by redundant agents

In all the cases the user will get only the current information so the problem of data inconsistency and non availability of data can be completely eradicated.

5. REMOTE DATA INTEGRITY

5.1 Introduction

The term remote data integrity refers to the accuracy and consistency of the stored data in the absence of any unauthorized access or modification at remote site. Remote data integrity is the only way to identify the integrity violations that possibly happens at remote site. Therefore this should be performed frequently to ensure safety of the remote data. RDI is not directly related to the security therefore it cannot be applied to provide data security in cloud. But it helps the user to identify any integrity violations at cloud data[31][32][33]. The user can go far any corrective measures upon the identification of any data outages. The advantage is this approach is that user need not download the data for integrity checking [34][35][36][37]. Thereby it considerably reduces the network bandwidth requirements. The drawback of the Remote data integrity algorithms is that they will only identify the integrity violations but fails to provide information about the location of the integrity breech and details of violation that happened at remote site. Therefore at the maximum the user can perform block level data dynamics and performing data level data dynamics is very difficult.

5.2. Remote Data Integrity: The Challenges

Checking the integrity of the cloud data is a complicate task for the reason that the data is not actually under the control of the owner and the service provider may have redundant copies of the same data possibly in different data centers. The possible solutions for integrity checking of remote data include the following;

a) Managing the integrity by self

b) Relying on interfaces provided by the cloud service provider

c) Relying on third party venders

Managing the integrity by self is extremely a complicated process because, intricacies of the cloud infrastructure are unknown and the user has to only rely upon the external interfaces provided by the service provider. The affordable solution is to keep the copies in the local system and perform integrity check by downloading the data from the cloud and compare it with local copies whenever necessary. It works well for a single user same time it completely closes avenues for data sharing opportunities. This also introduces new problems called bandwidth when the volume of the data to be checked is huge. Hence, it can't be practiced for organizations. Relying on the interface provided by the service provider is an alternate solution but it's very difficult to customize the interface to suit the users need. Third party auditing may provide solution but it may increases the cost factor and also content privacy is depends on the loyalty of the third party. The best way to overcome this problem is to provide configurable cloud interfaces to the user community to fit their needs.

6. CONCLUSION AND FUTURE WORK

Information security is the biggest thrust area in the world of present day computing. Almost every industry has already shifted their business or at least thinking of moving towards cloud. Hence it is important to provide security for their data. This work focuses on unsolved research issues on four major domains of public cloud namely communication, virtualization, data distribution and remote data integrity. In this paper we have tried our best efforts for providing realistic insight in the said domains with the view to help research people to comprehend the unsolved informational security research issues in the deployment of cloud design. This paper also proposes possible solutions in the discipline of informational security in public cloud from both client and service provider perspective. Our future work focuses on implementing water proof security solutions for public cloud in the aforementioned domains and provides low cost security solutions to the customers without compromise in security.

REFERENCES

[1] Buyya, Rajkumar, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility." *Future Generation computer systems* 25, no. 6 (2009): 599-616.

[2] Cachin, Christian, Idit Keidar, and Alexander Shraer. "Trusting the cloud." *Acm Sigact News* 40, no. 2 (2009): 81-86.

[3] Haoyong Lv and Yin Hu, "Analysis and Research about Cloud Computing Security Protect Policy", IEEE, 2011, pp. 214-216.

[4] Nagaraju Kilari, Dr R. Sridaran. "A survey on security threats for cloud computing." *International Journal of*

Engineering Research and Technology. Vol. 1. No. 7 (September-2012). ESRSA Publications, 2012.

[5] Alomary, ALAUDDIN Y., and M. S. Jamil. "New trends on security infrastructure for computer networks." In *Information and Communication Technologies: From Theory to Applications, 2004. Proceedings. 2004 International Conference on*, pp. 73-74. IEEE, 2004.

[6] Bishop, Matt. "What is computer security?." *Security & Privacy, IEEE* 1, no. 1 (2003): 67-69.

[7] Barnes, Jody. "Wired Network Security: Hospital Best Practices." (2006).

[8] Aman Bakshi and Yogesh B, "Securing cloud from DDOS Attacks using Intrusion Detection System in VM", IEEE, 2010, pp. 260-264.

[9] Courtois, Nicolas T., and Willi Meier. "Algebraic attacks on stream ciphers with linear feedback." In *Advances in Cryptology—EUROCRYPT 2003*, pp. 345-359. Springer Berlin Heidelberg, 2003.

[10] Arnaud de Borchgrave, Frank J. Cilluffo, Sharon L. Cardash, "Cyber Threats and Information Security: *Meeting the 21st Century Challenge,". Center for Strategic & Int'l Studies, May 2001.*

[11] Zhao, Weiming, Zhenlin Wang, and Yingwei Luo. "Dynamic memory balancing for virtual machines." *ACM SIGOPS Operating Systems Review* 43, no. 3 (2009): 37-47.

[12] Barham, Paul, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. "Xen and the art of virtualization." *ACM SIGOPS Operating Systems Review* 37, no. 5 (2003): 164-177.

[13] Huber, Nikolaus, Marcel von Quast, Michael Hauck, and Samuel Kounev. "Evaluating and Modeling Virtualization Performance Overhead for Cloud Environments." In *CLOSER*, pp. 563-573. 2011.

[14] Ray, Edward, and Eugene Schultz. "Virtualization security." In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, p. 42. ACM, 2009.

[15] Waldspurger, Carl A. "Content-based, transparent sharing of memory units." U.S. Patent 6,789,156, issued September 7, 2004.

[16] Miłós, Grzegorz, Derek G. Murray, Steven Hand, and Michael A. Fetterman. "Satori: Enlightened page sharing." In *Proceedings of the 2009 conference on USENIX Annual technical conference*, pp. 1-1. 2009.

[17] Kjos, Todd, Jonathan Ross, and Christophe De Dinechin. "Memory addressing for a virtual machine implementation on a computer processor supporting virtual hash-page-table searching." U.S. Patent 6,895,491, issued May 17, 2005.

[18] Banerjee, Ishan, Fei Guo, Kiran Tati, and Rajesh Venkatasubramanian. "Memory overcommitment in the ESX server." *VMware Technical Journal* 2 (2013).

[19] Arcangeli, Andrea, Izik Eidus, and Chris Wright. "Increasing memory density by using KSM." In *Proceedings of the linux symposium*, pp. 19-28. 2009.

[20] Waldspurger, Carl A. "Memory resource management in VMware ESX server."*ACM SIGOPS Operating Systems Review* 36, no. SI (2002): 181-194.

[21] VMware, E. S. X. "Understanding Memory Resource Management in VMware ESX 4.1."

[22] Harvey, T. U. C. H., Craig Newell, and Cyprien Laplace. "Cooperative memory resource management for virtualized computing devices." U.S. Patent 8,738,868, issued May 27, 2014.

[23] Li, Chin-Hung. "Evaluating the effectiveness of memory overcommit techniques on kvm-based hosting platform." In *Proceedings of World Academy of Science, Engineering and Technology*, no. 70. World Academy of Science, Engineering and Technology, 2012.

[24] Randell, Brian, and John E. Dobson. "Reliability and security issues in distributed computing systems." In *Symposium on Reliability in Distributed Software and Database Systems*, pp. 113-118. 1986.

[25] Özsu, M. Tamer, and Patrick Valduriez. *Principles of distributed database systems*. Springer Science & Business Media, 2011.

[26] Rushby, John M., and Brian Randell. "A distributed secure system." In *2012 IEEE Symposium on Security and Privacy*, pp. 127-127. IEEE Computer Society, 1983.

[27] Firdhous, Mohamed. "Implementation of Security in Distributed Systems-A Comparative Study." *arXiv preprint arXiv:1211.2032* (2012).

[28] Shen, Zhidong, and Xiaoping Wu. "The protection for private keys in distributed computing system enabled by trusted computing platform." In *Computer Design and Applications (ICCDA), 2010 International Conference on*, vol. 5, pp. V5-576. IEEE, 2010.

[29] Tiwari, Sanjay Kumar, Awadhesh Kumar Sharma, and Vishnu Swaroop. "Issues in Replicated data for Distributed Real-Time Database Systems."*IJCSIT) International Journal of Computer Science and Information Technologies* 2, no. 4 (2011): 1364-1371.

[30] Mazilu, Marius Cristian. "Database Replication." *Database Systems Journal* 1, no. 2 (2010): 33-38.

[31] Imran, Muhammad, et al. "Provenance based data integrity checking and verification in cloud environments." *PloS one*12.5 (2017): e0177576.

[32] Li, Jiguo, Hao Yan, and Yichen Zhang. "Certificateless public integrity checking of group shared data on cloud storage." *IEEE Transactions on Services Computing* (2018).

[33] Peng, Su, et al. "Efficient, dynamic and identity-based Remote Data Integrity Checking for multiple replicas." *Journal of Network and Computer Applications* 134 (2019): 72-88.

[34] Pir, Rana M. "Data Integrity Verification in Cloud Storage without using Trusted Third Party Auditor." (2014).

[35] Karthik. P and Krishna Kumar. V. Article: A Secure Access Code Technique for Remote Data Integrity on Public Cloud. International Journal of Computer Applications 77(14):26-31, September 2013.

[36] F. Sebe, J. Domingo-Ferrer, a. Martinez-Balleste, Y. Deswarte, and J. -J. Quisquarter, " Efficient Data Possession Checking in Critical information infrastructures", IEEE transactions on knowledge on Knowledge Engineering, Vol 20, pp. 1034-1038, Aug-2008.

[37] P Karthik and S Bhuvaneswari. Article: Design of Distributed Multi Agent Servers for Efficient Treatment of Health Care Issues. IJCA Proceedings on National Conference on Future Computing 2014 NCFC 2014(2):27-30, January 2014.