IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 4, Ver. V (Jul – Aug. 2014), PP 39-56
www.iosrjournals.org

Security Issues and Privacy in Cloud Computing

K.Durkesh[1], Ms.J.R.Thresphine[2]
[1]*(MTECH-CSE, Prist University, Puducherry)*
[2]*(A.P, Dept of CSE, Prist University, Puducherry)*

Abstract: *Recent advances have given rise to the popularity and success of cloud computing. However, when outsourcing the data and business application to a third party causes the security and privacy issues to become a critical concern. Throughout the study at hand, the authors obtain a common goal to provide a comprehensive review of the existing security and privacy issues in cloud environments. We have identified five most representative security and privacy attributes (i.e., confidentiality, integrity, availability, accountability, and privacy-preservability). Beginning with these attributes, we present the relationships among them, the vulnerabilities that may be exploited by attackers, the threat models, as well as existing defense strategies in a cloud scenario. Future research directions are previously determined for each attribute.*

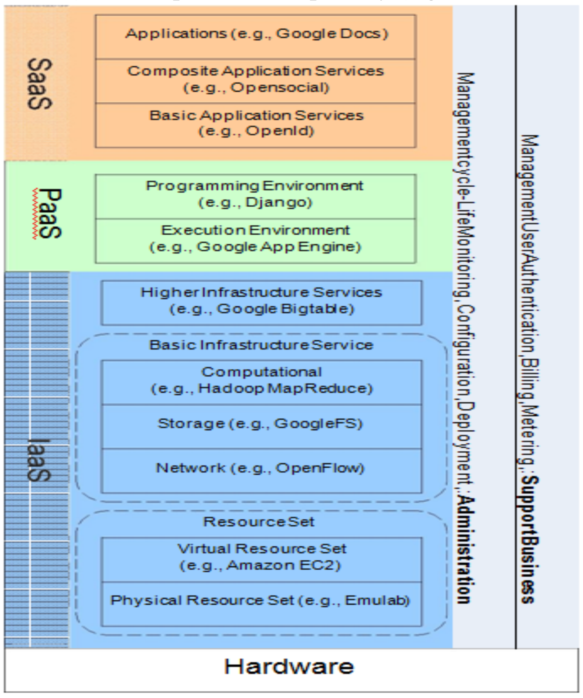
Index Terms: *cloud computing, security, privacy, trust, confidentiality, integrity, accountability, availability.*

I. Introduction

Cloud computing has begun to emerge as a hotspot in both industry and academia; It represents a new business model and computing paradigm, which enables on-demand provisioning of computational and storage resources. Economic benefits consist of the main drive for cloud computing due to the fact that cloud computing offers an effective way to reduce capital expenditure (CapEx) and operational expenditure (OpEx). The definition of cloud computing has been given in many literatures [1], [2], [3], [10], but nothing has gained wide recognition. Throughout this working text, we cite [1], which defines cloud computing as: ”*A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet.*”

A. Cloud architecture

Fig. 1 depicts the general architecture of a cloud platform, which is also called cloud stack [61]. Building upon hardware facilities (usually supported by modern data centers), cloud services may be offered in various forms from the bottom layer to top layer. In the cloud stack, each layer represents one service model. Infrastructure-as-a-Service (IaaS) is offered in the bottom layer, where resources are aggregated and managed physically (e.g., Emulab) or virtually (e.g., Amazon EC2), and services are delivered in forms of storage (e.g., GoogleFS), network (e.g., Openflow), or computational capability (e.g.,


SaaS
Applications (e.g., Google Docs)
Composite Application Services (e.g., Opensocial)
Basic Application Services (e.g., OpenId)
PaaS
Programming Environment (e.g., Django)
Execution Environment (e.g., Google App Engine)
IaaS
Higher Infrastructure Services (e.g., Google Bigtable)
Basic Infrastructure Service
Computational (e.g., Hadoop MapReduce)
Storage (e.g., GoogleFS)
Network (e.g., OpenFlow)
Resource Set
Virtual Resource Set (e.g., Amazon EC2)
Physical Resource Set (e.g., Emulab)
Hardware
Managementcycle-LifeMonitoring,Configuration,Deployment,:Administration
ManagementUserAuthentication,Billing,Metering,:SupportBusiness

Hadoop MapReduce). The middle layer delivers Platform-as-a-Service (PaaS), in which services are provided as an environment for programming (e.g., Django) or software execution (e.g., Google App Engine). Software as a Service (SaaS) locates in the top layer, in which a cloud provider further con-fines client flexibility by merely offering software applications as a service. Apart from the service provisioning, the cloud provider maintains a suite of management tools and facilities (e.g., service instance life-cycle management, metering and billing, dynamic configuration) in order to manage a large cloud system.

B. Cloud Characteristics and Security Challenges

The Cloud Security Alliance has summarized five essential characteristics [6] that illustrate the relation to, and differences from, traditional computing paradigm.

- **On-demand self-service** – A cloud customer may uni-laterally obtain computing capabilities, like the usage of various servers and network storage, as on demand, without interacting with the cloud provider.

- **Broad network access** – Services are delivered across the Internet via a standard mechanism that allows customers to access the services through heterogeneous thin or thick client tools (e.g., PCs, mobile phones, and PDAs).

- **Resource pooling** – The cloud provider employs a multi-tenant model to serve multiple customers by pooling computing resources, which are different physical and virtual resources dynamically assigned or reassigned according to customer demand. Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.

- **Rapid elasticity** – Capabilities may be rapidly and elastically provisioned in order to quickly scale out or rapidly released to quickly scale in. From customers' point of view, the available capabilities should appear to be unlimited and have the ability to be purchased in any quantity at any time.

- **Measured service** – The service purchased by customers can be quantified and measured. For both the provider and customers, resource usage will be monitored, controlled, metered, and reported.

Cloud computing becomes a successful and popular business model due to its charming features. In addition to the benefits at hand, the former features also result in serious cloud-specific security issues. The people whose concern is the cloud security continue to hesitate to transfer their business to cloud. Security issues have been the dominate barrier of the development and widespread use of cloud computing. There are three main challenges for building a secure and trustworthy cloud system:

- **Outsourcing** – Outsourcing brings down both capital expenditure (CapEx) and operational expenditure for cloud customers. However, outsourcing also means that customers physically lose control on their data and tasks. The loss of control problem has become one of the root causes of cloud insecurity. To address outsourcing security issues, first, the cloud provider shall be trustworthy by providing trust and secure computing and data storage; second, outsourced data and computation shall be verifiable to customers in terms of confidentiality, integrity, and other security services. In addition, outsourcing will potentially incur privacy violations, due to the fact that sensitive/classified data is out of the owners' control.

- **Multi-tenancy** – Multi-tenancy means that the cloud platform is shared and utilized by multiple customers. Moreover, in a virtualized environment, data belonging to different customers may be placed on the same physical machine by certain resource allocation policy. Adversaries who may also be legitimate cloud customers may exploit the co-residence issue. A series of security issues such as data breach [5], [17], [29], computation breach [5], flooding attack [26], etc., are incurred. Although Multi-tenancy is a definite choice of cloud venders due to its economic efficiency, it provides new vulnerabilities to the cloud platform. Without changing the multi-tenancy paradigm, it is imperative to design new security mechanisms to deal with the potential risks.

- **Massive data and intense computation** – cloud computing is capable of handling mass data storage and intense computing tasks. Therefore, traditional security mechanisms may not suffice due to unbearable computation or communication overhead. For example, to verify the integrity of data that is remotely stored, it is impractical to hash the entire data set. To this end, new strategies and protocols are expected.

C. Supporting techniques

Cloud computing has leveraged a collection of existing techniques, such as Data Center Networking (DCN), Virtualization, distributed storage, MapReduce, web applications and services, etc.

Modern data center has been practically employed as an effective carrier of cloud environments. It provides massive computation and storage capability by composing thousands of machines with DCN techniques.

Virtualization technology has been widely used in cloud computing to provider dynamic resource allocation and service provisioning, especially in IaaS. With virtualization, multiple OSs can co-reside on the same physical machine without interfering each other.

MapReduce [53] is a programming framework that sup-ports distributed computing on mass data sets. This breaks large data sets down into small blocks that are distributed to cloud servers for parallel computing. MapReduce speeds up the batch processing on massive data, which makes this be-come the preference of computation model for cloud venders.

Apart from the benefits, the former techniques also present new threats that have the capability to jeopardize cloud security. For instance, modern data center suffers bandwidth under-provisioning problems [24], which may be exploited and may consequently perform a new DOS attack [20] due to the shared infrastructure in cloud environments. Virtual Machine (VM) technique also has the capability to enable adversaries to perform cross-VM attacks [17] and timing attacks [60] due to VM co-residence. Further details are to be discussed in Section II and Section III.

D. Attribute-driven Methodology

Security and privacy issues in cloud environments have been studied and surveyed in prior literatures. To better understand these issues and their connections, researchers have employed various criteria to build a comprehensive picture.

Gruschka et al. [32] suggests modeling the security ecosystem based on the three participants of cloud system: service user, service instance, and the cloud provider. The authors classify the attack into six categories (i.e., user to service, service to user, user to cloud, cloud to user, service to cloud,

and cloud to service). This taxonomy is suggested to describe the threats and vulnerabilities presented in cloud computing. Subashini et al. [39] have summarized the cloud security issues based on service delivery models (i.e., IaaS, PaaS, SaaS). Vaquero et al. [4] have given another comprehensive survey according to the seven main threats presented in [7]. Grobauer et al. [70] have pointed out the importance to distinguish general security issues from cloud-specific security issues. In addition, cloud computing vulnerabilities are discussed and summarized from various aspects.

In this paper, we consider the cloud environment as a new computing platform to which the classic methodology of security research can be applied as well. Therefore, we determine to employ an attribute-driven methodology to con-duct our review. We employ the ecosystem of cloud security and privacy in view of five security/privacy attributes (i.e., confidentiality, integrity, availability, accountability, and privacy-preservability), shown in Fig. 2, that are the most representative ones in current research advances. Some re-searchers regard privacy as one component of security, while in this paper, we separate privacy from security due to its importance and specialty in cloud environments. Privacy is considered as highly relevant to security, as well as other security attributes that have positive or negative influences on privacy. The security ecosystem is generic and is applicable to any computer and networked systems. In order to build a comprehensive picture of cloud security and privacy, we follow the semantics of the ecosystem to organize this survey. Starting with the five attributes, we have discussed the vulnerabilities that can be exploited by adversaries. We have also surveyed the threat models that attackers can use to jeopardize the security objectives. Some threats and attacks have been undertaken properly, while others are still remaining to be solved. These aspects are also related security and privacy in systems other than cloud computing [99], [100], [101], [102], [103], [104], [105], [106], [107], [108], [109], [110], [111], [112], [113], [114], [115], [116], [117], [118], [119], [120], [121], [122], [123].

E. Notation System

1. To better demonstrate the connection among vulnerability, threat, and defense mechanism. We employ the following notation system: let Vi denote a type of vulnerability, Ti.j
2. denote a type of threat that takes advantage of Vi, and Di.j.k denote a defense mechanism that deals with Ti.j . For instance, vulnerability V1 may be exploited by adversaries in order to create a threat model T1.1,

which shall be patched by security solution D1.1.1.

F. Cloud Vulnerabilities

1) V_1 – VM co-residence: In cloud computing, co-residence (or co-tenancy) means that multiple independent customers share the same physical infrastructure. Concretely, virtual machines belonging to different customers may be placed in the same physical machine. VM co-residence has raised certain security issues, such as Cross-VM attack [17] and Malicious SysAdmin [60].

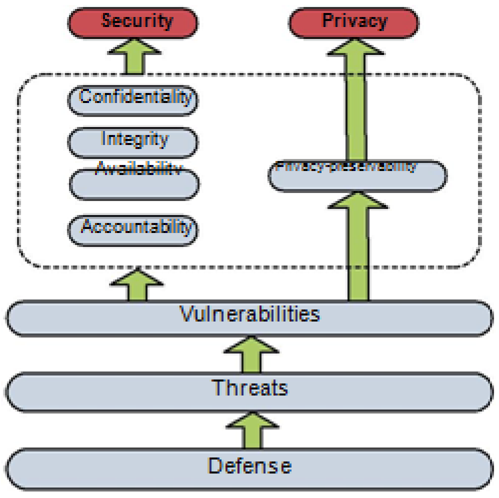


Fig. 2. Ecosystem of Cloud Security and Privacy

2) V_2 – Loss of Physical Control: Cloud customers have their data and program outsourced to cloud servers. As a result, owners lose direct control on the data sets and programs. Loss of physical control means that customers are unable to resist certain attacks and accidents. For example, data or software may be altered, lost, or even deleted; in addition, it is difficult and impractical to ensure data/computation integrity and confidentiality with traditional methods.

3) V_3 – Bandwidth Under-provisioning: A traditional DOS/DDOS attack does exist in cloud computing, and relative solutions have been given in prior researches [21], [22]. Specific to cloud computing, there is a new type of DOS attack that takes advantage of the current under-provisioned cloud-computing infrastructure. According to Cisco's design guide [24], a data center is usually designed to be under-provisioned with a factor of 2.5:1 to 8:1, meaning that the actual network capacity is much less than the aggregate capacity of the hosts located in the same subnet.

4) V_4 – Cloud Pricing Model: Cloud computing adheres to the pay-as-you-go pricing model [10] that determines the cost of services in terms of metrics such as server hours, band-width, storage, etc. Since all cloud customers are financially responsible for the services they use, attackers always have incentives to harass the billing process by exploiting the pricing model. For example, Economic Denial of Sustainability (EDoS) attack [19] manipulates the utility pricing model and causes unmanageable costs for cloud customers.

The remainder of this paper is structured as follows: Sections II to VI discuss confidentiality, integrity, availability, accountability, and privacy in cloud computing, respectively; finally, the paper is concluded in Section VII.

II. Cloud Confidentiality

When dealing with cloud environments, confidentiality im-plies that a customer's data and computation tasks are to be kept confidential from both the cloud provider and other customers. Confidentiality remains as one of the greatest concerns with regards to cloud computing. This is largely due to the fact that customers outsource their data and computation tasks on cloud servers, which are controlled and managed by potentially untrustworthy cloud providers.

A. Threats to Cloud Confidentiality

1) $T_{1.1}$ – Cross-VM attack via Side Channels: Ristenpart et al. [17] demonstrates the existence of Cross-VM attacks in an Amazon EC2 platform. A Cross-VM attack exploits the nature of multi-tenancy, which enables that VMs belonging to different customers may co-reside on the same physical machine. Aviram et al. [60]

regard timing side-channels as an insidious threat to cloud computing security due to the fact that a) the timing channels pervasively exist and are hard to control due to the nature of massive parallelism and shared infrastructure; b) malicious customers are able to steal information from other ones without leaving a trail or raising alarms. There are two main steps to practically initiate such an attack:

- **Step 1: placement**. An adversary needs to place a malicious VM on the physical server where the target client's VM is located. To achieve this, an adversary should first determine where the target VM instance is located; this can be done with network probing tools such as nmap, hping, wget, etc. An adversary should also be able to determine if there are two VM instances; 1) comparing Domain0's IP addresses to see if they match, and 2) measuring the small packet round-trip time can do this check. The correctness of co-resident checks can be verified by transmitting messages between instances via a covert channel. After all the prep work, a malicious VM instance must be created on the target physical machine by specifying a set of parameters (e.g., zone, host type); there are two basic strategies to launch such a VM: 1) brute-force strategy, which simply launches many instances and checks co-residence with the target; 2) an adversary can exploit the tendency that EC2 launches new instances on the same small set of physical machines. The second strategy takes advantage of EC2's VM assigning algorithm by starting a malicious VM after a victim VM is launched so that they will likely be assigned to the same physical server; this approach surely has better success rate of placement.

- **Step 2: extraction**. After step 1, a malicious VM has co-resided with the victim VM. Since the malicious VM and the victim are sharing certain physical resources, such as data cache, network access, CPU branch predicators, CPU pipelines, etc., there are many ways an adversary can employ attacks: 1) measuring a cache usage that can estimate the current load of the server; 2) estimating a traffic rate that can obtain the visitor count or even the frequently requested pages; 3) a keystroke timing attack that can steal a victim's password by measuring time between keystrokes.

As follow-up work, various covert channels are investigated and in-depth analysis is provided. Attackers can easily exploit L2 cache, due to its high bandwidth. Xu et al. have particularly explored the L2 cache covert channel with quantitative assessment [71]. It has been demonstrated that even the channel bit rate is higher than the former work [17], the channel's ability to exfiltrate useful information is still limited, and it is only practical to leak small secrets such as private keys. Okamura et al. developed a new attack, which demonstrates that CPU load can also be used as a covert channel to encode information [72]. Memory disclosure attack [81], [82] is another type of cross-VM attack. In a virtualized environment, memory reduplication is a technique to reduce the utilization of physical memory by sharing the memory pages with same contents. A memory disclosure attack is capable of detecting the existence of an application or a file on a co-residing VM by measuring the write access time that differs between reduplicated pages and regular ones.

2) $T_{1.2}$ – Malicious SysAdmin: The Cross-VM attack discusses how others may violate confidentiality cloud customers that co-residing with the victim, although it is not the only threat. Privileged sysadmin of the cloud provider can perform attacks by accessing the memory of a customer's VMs. For instance, Xenaccess [30] enables a sysadmin to directly access the VM memory at run time by running a user level process in Domain0.

B. Defense Strategies

Approaches to address cross-VM attack fall into six categories: a) placement prevention intends to reduce the success rate of placement; b) physical isolation enforcement [80]; c) new cache designs [75], [76], [77], [78]; d) fuzzy time intends to weaken malicious VM's ability to receive the signal by eliminating fine-grained timers [73]; e) forced VM determinism [60] ensures no timing or other non-deterministic information leaking to adversaries; f) cryptographic implementation of timing-resistant cache [79]. Since c), d), e), and f) are not cloud-specific defense strategies, we do not include details in this section.

1) $D_{1.1.1}$ – Placement Prevention: In order to reduce the risk caused by shared infrastructure, a few suggestions to defend the attack in each step are given in [17]. For instance, cloud providers may obfuscate co-residence by having Dom0 not respond in trace route, and/or by randomly assigning internal IP addresses to launched VMs. To reduce the success rate of placement, cloud providers might let the users decide where to put their VMs; however, this method does not prevent a brute-force strategy.

[80] $D_{1.1.2}$ – Co-residency Detection: The ultimate solution of cross-VM attack is to eliminate co-residency. Cloud customers (especially enterprises) may require physical isolation, which can even be written into the Service Level Agreements (SLAs). However, cloud vendor may be reluctant to abandon virtualization that is beneficial to cost saving and resource utilization. One of the left options is to share the infrastructure only with

”friendly” VMs, which are owned by the same customer or other trustworthy customers. To ensure physical isolation, a customer should be enabled to verify its VMs’ exclusive use of a physical machine. Home Alone is a system that detects co-residency by employing a side-channel (in the L2 memory cache) as a detection tool. The idea is to silence the activity of ”friendly” VMs in a selected portion of L2 cache for a certain amount of time, and then measure the cache usage to check if there is any unexpected activity, which indicates that the physical machine is co-resided by another customer.

3) $D_{1.1.3}$ – No Hype: No Hype ([83], [84]) attempts to minimize the degree of shared infrastructure by removing the hypervisor while still retaining the key features of virtualization. The No Hype architecture provides a few features: i) the ”one core per VM” feature prevents interference between VMs, eliminates side channels such as L1 cache, and retains multi-tenancy, since each chip has multiple cores; ii) memory partition restricts each VM’s memory access on a assigned range; iii) dedicated virtual I/O devices enables each VM to be granted direct access to a dedicated virtual I/O device. No-Hype has significantly reduced the hypervisor attack surface, and increased the level of VM isolation. However, No Hype requires to change hardware, making it less practical when consider applying it to current cloud infrastructures.

4) $D_{1.2.1}$ – Trusted Cloud Computing Platform: Santos et al. [29] present a trusted cloud-computing platform (TCCP), which offers a closed box execution environment for IaaS services. TCCP guarantees confidential execution of guest virtual machines. It also enables customers to attest to the IaaS provider and to determine if the service is secure before their VMs are launched into the cloud.

The design goals of TCCP are: 1) to confine the VM execution inside the secure perimeter; 2) that a sysadmin with root privileges is unable to access the memory of a VM hosted in a physical node. TCCP leverages existing techniques to build trusted cloud computing platforms. This focuses on solving confidentiality problems for clients’ data and for computation outsourced to the cloud. With TCCP, the sysadmin is unable to inspect or tamper with the content of running VMs.

5) Other opinions: retaining data control back to customer:

Considering the customer’s fear of losing the data control in cloud environments, Descher et al. [40] propose to retain data control for the cloud customers by simply storing encrypted VMs on the cloud servers. Encrypted VM images guarantee rigorous access control since only the authorized users known as key-holders are permitted access. Due to the encryption, the data cannot be mounted and modified within the cloud without an access key, assuring the confidentiality and integrity. This approach offers security guarantees before a VM is launched; however, there are ways to attack the VM during running time
[30] and to jeopardize the data and computation.

C. Summary and Open issues

Regarding confidentiality, cross-VM attack and malicious SysAdmin mainly threaten a cloud system; both threats take advantage of the vulnerability of virtualization and co-residence. Other tenants perform cross-VM attack, whereas the malicious SysAdmin is inside attack from cloud vender. Defending these threats is not a trivial task due to the following facts: 1) various side channels and other shared components can be exploited, and defending each of them is not an easy job; 2) There are a few open issues to be explored:

• Co-residency detection is considered as a promising technique since customers should be able to check whether the physical isolation is well enforced. Home Alone [80] has the ability to achieve accuracy of detection on L2 cache side channels. However, besides L2 cache, other side channels may be exploited as well. Therefore, in order to provide thorough detection of co-residence, a suite of detection methods targeting on various side channels should be developed.

• NoHype has opened another window to deal with cross-VM threat. However, current commodity hardware imposes limitations to implement NoHype. Additionally, live VM migration is not well supported by this new architecture. Therefore, before making a real step for-ward, researchers need to address the hardware changes to accommodate NoHype and to maintain more features for VM management.

III. CLOUD INTEGRITY

Similar to confidentiality, the notion of integrity in cloud computing concerns both data integrity and computation integrity. Data integrity implies that data should be honestly stored on cloud servers, and any violations (e.g., data is lost, altered, or compromised) are to be detected. Computation integrity implies the notion that programs are executed without being distorted by malware, cloud providers, or other malicious users, and that any incorrect computing will be detected.

A. Threats to Cloud Integrity

1) $T_{2.1}$ – data loss/manipulation: In cloud storage, applications deliver storage as a service. Servers keep large amounts of data that have the capability of being accessed on rare occasions. The cloud servers are distrusted in terms of both security and reliability [14], which means that data may be lost or modified maliciously or accidentally. Administration errors may cause data loss (e.g., backup and restore, data migration, and changing memberships in P2P systems [11]). Additionally, adversaries may initiate attacks by taking advantage of data owners' loss of control over their own data.

2) $T_{2.2}$ – dishonest computation in remote servers: With outsourced computation, it is difficult to judge whether the computation is executed with high integrity. Since the computation details are not transparent enough to cloud customers, cloud servers may behave unfaithfully and return incorrect computing results; they may not follow the semi-honest model. For example, for computations that require large amount of computing resources, there are incentives for the cloud to be ”lazy” [85]. On the other hand, even the semi-honest model is followed, problems may arise when a cloud server uses outdated, vulnerable code, has misconfigured policies or service, or has been previously attacked with a rootkit, triggered by malicious code or data [86].

B. Defense Strategies

To tackle the challenges of integrity checking presenting in cloud storage. The main challenge of integrity checking is that tremendous amounts of data are remotely stored on untrustworthy cloud servers; as a result, methods that require hashing for the entire file become prohibitive. In addition, it is not feasible to download the file from the server and perform an integrity check due to the fact that it is computationally expensive as well as bandwidth consuming. Each of the former notions is not acceptable in cloud environments.

Provable Data Possession, referred to as (PDP) [11], [12], [14], [15], becomes employed through the process of checking the data integrity with cloud storage in order to answer the question, ”Is it possible for customers to be sure that the outsourced data is honestly stored in cloud?”

a) A Naive Method

For comparison purposes, a naive method is given in [12]. This idea consists of the client computing a hash value for file F with a key k (i.e., h(k, F)) and subsequently sending F to the server. Once the client finds a necessity to check the file, it releases k and sends k to the server, which is subsequently asked to re-compute the hash value, based on the F and k; after this, the server replies to the client with the hash result for comparison. The client can initiate multiple checks by keeping different keys and hash values. This approach provides strong proof that the server still retains F . However, the negative aspect is the high overhead that is produced. This overhead exists because each time of verification requires the server to run a hashing process over the entire file. The notion at this moment is computationally costly, even for lightweight hashing operations.

b) Original Provable Data Possession (PDP)

The original PDP model [11] requires that the data is pre-processed in the setup phase in order to leave some meta-data on the client side for verification purposes subsequently, for that data to be sent to the cloud server. Once the client feels a necessity to check the data integrity at a later time, he/she sends a challenge to the cloud server, which will respond with a message based on the data content. After combining the reply and the local meta-data, the client is able to prove whether the integrity of the data is violated. The probabilistic guarantee of PDP shows that PDP can achieve a high probability for detecting server misbehavior with low computational and storage overhead.

PDP is only applicable to static files (i.e., append-only files), meaning that the data may not be changed once uploaded to the server. This limitation reduces its applicability to cloud computing due to the fact that it is featured with dynamic data management.

c) Proof of Irretrievability (POR)

Proof of Irretrievability (PoR) [12] employs an auditing protocol when solving a similar problem to PDP. The problem is that each of the two enables clients to check the integrity of outsourced data without having to retrieve it. PoR is designed to be lightweight. In other words, it attempts to minimize the storage in client and server side, the communication complexity of an audit, and the number of data-blocks accessed during an audit [12].

The POR protocol is depicted in Fig. 3. The user stores only a key, which is used to encode the file F in order to get

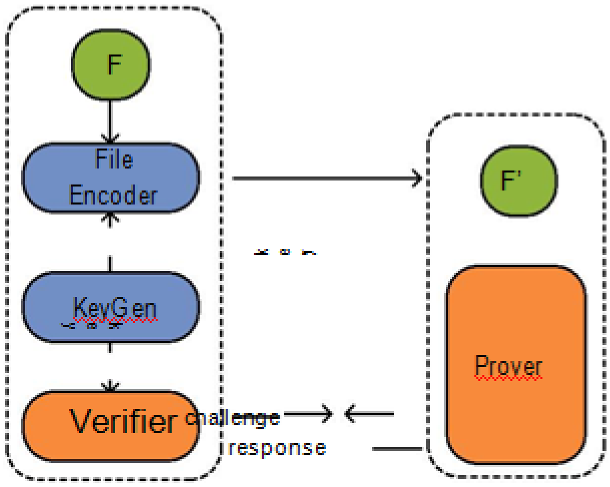


Fig. 3. Schematic of POR System

the encrypted file F'. The task is that a set of sentinel values are embedded into F', and the server only stores F' without knowing where the sentinels may be since the sentinels are indistinguishable from regular data blocks. In the challenge and response protocol, the server is asked to return a certain subset of sentinels in F'. If the server has tampered with or deleted F', there is high probability that certain sentinels are also corrupted or lost; this causes the server to be unable to generate a complete proof for the original file. Therefore, a client has evidence to prove that the server has corrupted the file. Due to the fact that the number of sentinels is limited, POR adopts error-correcting codes to recover the file with only a small fraction being corrupted.

Similar to PDP, PoR can only be applied to static files. A subtle change to the file will ruin the protocol and completely confuse the clients.

d) Scalable PDP

Scalable PDP [14] is an improved version of the original PDP. The difference in the two are described as the following: 1) scalable PDP adopts symmetric key encryption instead of public-key to reduce computation overhead, but scalable PDP does not support public verification due to symmetric encryption; 2) scalable PDP has added dynamic operations on remote data. One limitation of scalable PDP is that all challenges and answers are pre-computed, and the number of updates is limited and fixed as a priori.

e) Dynamic PDP

The goal of Dynamic PDP (DPDP) [15] is to support full dynamic operations (e.g., append, insert, modify, and delete). The purpose of dynamic operations is to enable authenticated insert and delete functions with rank-based authenticated di-rectories that are built on a skip list. The experiment result shows that, although the support of dynamic updates costs certain computational complexity, DPDP is practically efficient. For instance, to generate proof for a 1GB file, DPDP only produces 415 KB proof data and 30 ms computational overhead.

TABLE I
APPROACHES OF DATA INTEGRITY CHECKING IN CLOUD STORAGE

Keywords	**Protocol Sketch**	**Pros**	**Cons**	**Primitives**
PDP [11]	1. Client: KeyGen 2. Client: TagBlock 3. Server: GenProof 4. Client: CheckProof	- Supports both encrypted data and plain data. - Efficient: only a small portion of file needs to be accessed to generate proof on the server. - Offers public verifiability.	- Only supports integrity check-ing for static data (i.e., append only). - Probabilistic guarantees may re-sult in false positive.	Homomorphic hashing: to com-pose multiple block inputs into a single value to reduce the size of proof.
POR [12]	1. Client: KeyGen 2. Client: Encode 3. Server: Extract	- Ability to recover file with error-correcting code. - Efficient.	- Static data only. - File needs to be encrypted be-fore uploading to server.	Error-correcting code: to recover a partially corrupted file.

	4. Client: Challenge 5. Server: Respond 6. Client: Verify		- Needs additional space to hide sentinels in.	
Scalable PDP [14]	1. Client: KeyGen 2. Client: TokenGen 3. Server: Update 4. Client: Challenge 5. Server: Proof 6. Client: Verify	- No bulk encryption is required. - Allow outsourcing dynamic data in some degree. - Rely on symmetric-key which is more efficient than public-key encryption.	- Does not offer public verifiabil- ity. - All challenges and answers are pre-computed. - Number of updates is limited and fixed as a priori.	- Symmetric-key cryptography. - Message Authentication Code (MAC).
Dynamic PDP [15]	1. Client: KeyGen. 2. Client: PrepUpdate Server: 3. PerfromUpdate 4. Client: VerifyUpdate 5. Client: Challenge 6. Server: Proof 7. Client: Verify	- Support fully dynamic data op- eration (i.e., insert, modification, delete, and append). - All challenges and answers are dynamically generated.	- Fully dynamic support causes relatively computational higher , communication, and storage overhead.	- Rank-based authenticated direc- tory. - RSA-tree. - Authenticated skip list.
HAIL [16]	1. Client: KeyGen 2. Client: Encode 3. Server: Decode 4. Client: Challenge 5. Server: Respond 6. Client: Verify 7. Svr / cli: Redistribute	- Ability to check integrity in dis- tributed storage via data redun- dancy. - Proof is compact in size and is independent of data size.	- Static data only.	- Pseudorandom functions. - Message authentication codes (MACs). - Universal hash functions.

Perform Update is run by a server to perform the actual file update, and subsequently returns an update proof to the client who, in turn, verifies the server behavior during the update.

f) **HAIL:** A High-Availability and Integrity Layer for cloud storage

HAIL [16] differs from the prior work with regards to the fact that it considers a distributed setting in which a client must spread a file across multiple servers with redundancy and only store a small constant state in local machine. The main threats that HAIL combats are mobile adversaries, which may possibly corrupt file F by undermining multiple servers.

g) Summary of PDP

PDP is a class of problems that provides efficient and practical approaches in order to determine whether the outsourced data is honestly stored. We have summarized and compared several newly emerging approaches in Table I. The evolution of PDP shows the improvements from static data to dynamic data as well as from single-server setting to distributed-servers setting.

2) $D_{2.1.2}$ – Third Party Auditor: Instead of letting customers verify data integrity, it is also possible to offload task of integrity checking to a third party which can be trusted by both cloud provider and customers. Wang et al. [45] propose to adopt a third party auditor (TPA) to check the integrity of outsourced data in cloud environments. TPA ensures the following: 1) cloud data can be efficiently audited without a local data copy, and cloud clients suffer no on-line overhead for auditing; 2) no new vulnerabilities will be introduced to jeopardize data privacy. The key technique is a public-based homomorphic authenticator, which has been utilized in existing literatures [46]. When combining a homomorphic authenticator with random masking, TPA becomes unable to access the data content while it is performing auditing.

3) Combating dishonest computing: The outsourcing feature of cloud computing motivates researchers to revisit a classic problem that addresses integrity of external computations. How many a machine outsource a computation to another machine and then, without running the computation locally, verify the correctness of the result output by the other machine [87]? Conventional strategies to check external computation integrity fall into

four categories:

- $D_{2.2.1}$: **Re-computation** requires the local machine to re-do the computation, and then compare the results. Re-computation guarantees 100% accuracy of mistake detection, and does not require trusting the cloud vendor. However, the cost is usually unbearable due to the fact that each of the verifications require at least the equal time as the original computation. To this end, customers could possibly have no incentive to verify computation integrity in this manner. A variation of re-computation is sampling [66], which offers probabilistic guarantees of mistake detection, depending on the degree of sampling. Sampling trades accuracy for efficiency.

- $D_{2.2.2}$: **Replication** assigns one computation task to multiple machines, and then compares the results. Majority voting may be employed to determine correctness. Replication assumes semi-trust to cloud vender because both computation and verification are conducted remotely.
Intelligent adversaries that control certain amounts of machines may bypass replication checking by returning the same incorrect results.

- $D_{2.2.3}$: **Auditing** [51], [89] usually works together with logging. During the execution of a computation, a logging component records all critical events into a log file, which is subsequently sent to one or multiple auditors for review. Auditing is a typical approach to do forensics investigation. One drawback of auditing is that if the attacker understands the computation better than the auditor, it is possible for the attacker to manipulate data bits without being detected.

- $D_{2.2.4}$: **Trusted Computing** [29], [58] enforces the computer to behave consistently in expected ways with hardware and software support. The key technique of integrity checking is known as remote attestation, which works by having the hardware generate a certificate stating that what software is running. The certificate can then be sent to the verifier to show that the software is unaltered. One assumption of trusted computing is that some component like the hardware and the hypervisor is not physically altered.

Some verification methods ([85], [90], [91], [92]) are do-main or application specific. For example, Wang et al. have designed a practical mechanism for linear programming (LP) outsourcing [85]; by exploring the duality theorem of LP computation and deriving the conditions that correct result must satisfy, the verification mechanism only incurs close-to-zero additional cost on both cloud servers and customers. Freivalds' method [89] verifies a $m * m$ matrix multiplication in $O(m^2)$ with a randomized algorithm. Blanton et al. [92] provide an approach of outsourcing large-scale biometric computation with reasonable overhead.

The above approaches either rely on various assumptions that have restrictions, or incur high computation cost. The ultimate goal, however, is to provide both practical and unconditional integrity verification for remote computation. Towards this goal, Setty et al. [95] suggest to seek help from some early research results such as interactive proof [93], and probabilistically checkable proofs (PCPs) [94], which attempt to design an ideal method that enables a customer to check an answer's correctness in constant time, with a suitably encoded proof and under a negligible chance of false positive [94]. To date, PCP-based verification method is not practical for general purpose yet, according to Setty's research. Although its application on a particular problem (i.e., matrix multiplication) seems to be encouraging, the author has also pointed out some serious issues such as expensive setup time.

C. Summary and Open issues

Cloud Integrity becomes vulnerable because the customers do not physically control their data and software. For data integrity, there are two challenges: i) huge data volume makes conventional hashing scheme not viable; ii) integrity checking can only be practical when there are additional requirements, which increase the difficulty; for instance, dynamic operation support on remote data is non-trivial with integrity guarantees, and in distributed setting, both integrity and consistency are taken into consideration. On the other hand, computation integrity is far tougher. The main challenge is the lack of knowledge of the computation internals; if the verification method applies to generic computations. A well-designed integrity checking method satisfies the following conditions: i) for practical concerns, the workload of local computation of verification should be less than the original computation. Otherwise it is not efficient to do outsourcing; ii) the proof can be verified by any party to ensure non-repudiation; iii) no or few assumption is imposed.

There is great research potential in this area:
- Combing integrity checking with other realistic requirements is a promising research trend. State-of-the-art researches have studied to support dynamic operation on cloud data; also, single machine setting is extended to

the distributed setting. However, no prior works investigate integrity checking along with both dynamic operation and distributed setting. Moreover, other traditional features on distributed system such as fault-tolerance can be considered as well.

• A significant advance will be made if a practical and unconditional verification method is developed for computation integrity. It is an important attempt [95] to reduce computation complexity when applying probabilistically checkable proofs to matrix multiplication as a case study. However, even in one case study, a few serious problems arise. The improvement space is still large.

• On the other hand, domain-specific method can achieve satisfying effects because these methods usually take advantage of the inner features of computation. Some scientific computations such as Linear Programming and matrix operation have their outsourcing versions. How-ever, other types such as non-linear optimization are remaining to be outsourced to cloud with strong integrity assurance.

IV. Cloud Availability

Availability is crucial since the core function of cloud computing is to provide on-demand service of different levels. If a certain service is no longer available or the quality of service cannot meet the Service Level Agreement (SLA), customers may lose faith in the cloud system. In this section, we have studied two kinds of threats that impair cloud availability.

A. Threats to Cloud Availability

1) $T_{3.1}$ – Flooding Attack via Bandwidth Starvation: In a flooding attack, which can cause Deny of Service (DoS), a huge amount of nonsensical requests are sent to a particular service to hinder it from working properly. In cloud computing, there are two basic types [34] of flooding attacks:

• **Direct DOS** – the attacking target is determined, and the availability of the targeting cloud service will be fully lost.
• **Indirect DOS** – the meaning is twofold: 1) all services hosted in the same physical machine with the target victim will be affected; 2) the attack is initiated without a specific target.

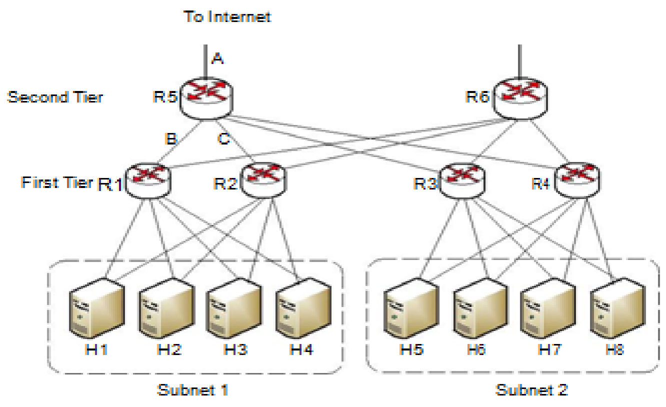


Fig. 4. Traditional Data Center Network Architecture

The authors in [34] also point out that one of the con-sequences of a flooding attack is that if a certain cloud service is unavailable or the quality of service is degraded, the subscribers of all affected services may need to continue paying the bill. However, we argue that since cloud providers must have previously signed a Service Level Agreement (SLA) with their clients, a responsible party must be determined once the service level is degraded to some threshold since clients will be aware of that degradation. We will elaborate upon this problem (i.e., cloud accountability) in the next section.

The nature of under-provisioning and public openness in a cloud system brings new vulnerability that can be exploited to carry out a new DOS attack to jeopardize the cloud service provision by saturating the limited network bandwidth. As shown in Fig. 4, links A, B, C are uplinks of router R5, R1, and R2, respectively. Suppose that link B is the active link and link C is the fail-over link (i.e., a link will be activated when the active link is down). Due to under-provisioning, the aggregate capacity of H1, H2, H3, and H4 (which form the subnet 1) is a few times larger than any capacity for links A, B, or C. In order to saturate link B, attackers (which may be a few hosts controlled by the adversary) in subnet 1 only need to generate enough traffic to target the hosts in another subnet (e.g., subnet 2). Once link B is saturated by the non-sense traffic, hosts in subnet1 are unable to deliver services to cloud users.

To initiate such a DOS attack (bandwidth starvation) effectively, there are a few steps:
1) **Topology identification** – Since only hosts in different subnets are connected by bottleneck links, an adversary needs to first identify the network topology. By exploiting the multiplexing nature of a router, the number of routers between two hosts can be determined; this helps selected hosts picture the topology.

2) **Gaining access to enough hosts** – The number of hosts to perform the attack is determined by the uplink's capacity, which can be estimated by some tools such as Pathload [26], Nettimer [27], or Bprobe [25].

3) **Carrying out the attack** – The author suggests employing UDP traffic because it will starve other TCP sessions.
2) $T_{4.1}$ – Fraudulent Resource Consumption (FRC) attack:

A representative Economic Denial of Sustainability (EDoS) attack is FRC [96], [97], which is a subtle attack that may be carried out over a long period (usually lasts for weeks) in order to take effect. In cloud computing, the goal of a FRC attack is to deprive the victim (i.e., regular cloud customers) of their long-term economic availability of hosting web contents that are publicly accessible. In other words, attackers, who act as legal cloud service clients, continuously send requests to website hosting in cloud servers to consume bandwidth, which bills to the cloud customer owning the website; seems to the web server, those traffic does not reach the level of service denial, and it is difficult to distinguish FRC traffic from other legitimate traffic. A FRC attack succeeds when it causes financial burden on the victim.

B. Defense strategy
1) $D_{3.1.1}$ – defending the new DOS attack: This new type of DOS attack differs from the traditional DOS or DDOS [24] attacks in that traditional DOS sends traffic to the targeting application/host directly while the new DOS attack does not; therefore, some techniques and counter-measures [21], [22] for handling traditional DOSs are no longer applicable.

A DOS avoidance strategy called service migration [20] has been developed to deal with the new flooding attack. A monitoring agent located outside the cloud is set up to detect whether there may be bandwidth starvation by constantly probing the cloud applications. When bandwidth degradation is detected, the monitoring agent will perform application migration, which may stop the service temporarily, with it resuming later. The migration will move the current application to another subnet of which the attacker is unaware. Experiment results show that it only takes a few seconds to migrate a stateless web application from one subnet to another.

2) $D_{4.1.1}$ – FRC attack detection: The key of FRC detection is to distinguish FRC traffic from normal activity traffic. Idziorek et al. propose to exploit the consistency and self-similarity of aggregate web activity [96]. To achieve this goal, three detection metrics are used: i) Zipf 's law [97] are adopted to measure relative frequency and self-similarity of web page popularity; ii) Spearman's footrule is used to find the proximity between two ranked lists, which determines the similarity score; iii) overlap between the reference list and the comparator list measures the similarity between the training data and the test data. Combining the three metrics yields a reliable way of FRC detection.

C. Summary and Open Issues
Service downgrade can be resulted by both internal and external threats. An internal threat comes from malicious cloud customers who take advantage of the bandwidth under-provisioning property of current DCN architecture to starve legitimate service traffic. On the other hand, external threat refers to the EDoS attack, which degrades the victim's long-term economic availability. cloud platform are novel and worthwhile to be investigated. The following issues could be future research directions:

- $D_{3.1.1}$ gives an avoidance strategy that adopts service migration. DoS avoidance is however not sufficient to entirely defend this attack, because the adversaries are not identified yet. The issue can be further addressed by accountability. In this case, to track the malicious behavior, the key is to identify the origin of non-sense traffic that tried to saturate the connection link.

- $D_{4.1.1}$ describes a mechanism for FRC detection. How-ever, it is not clear that how does a victim react to the attack, and the identification of attackers is not presented, as well. To complement D4.1.1, new researches in this area are expected.

• FRC attack is carried out by consuming bandwidth, which is one of the resources that are billable. However, other resources, such as computing capabilities and storage, are also potentially vulnerable to EDoS attack. Therefore, it is imperative to discover viable threat models and defense strategies towards a comprehensive study of EDoS attack.

V. Cloud Accountability

While accountability has been studied in other systems [124], [125], [126], [127], [128], [129], [130], [131], it is essential in order to build trust relationships in cloud environ-ment [35], [47], [48], [50], [52], [88]. Accountability implies that the capability of identifying a party, with undeniable evidence, is responsible for specific events [124], [125], [126], [127], [128], [129], [130], [131]. When dealing with cloud computing, there are multiple parties that may be involved; a cloud provider and its customers are the two basic ones, and the public clients who use applications (e.g., a web application) outsourced by cloud customers may be another party. A fine-grained identity, however, may be employed to identify a specific machine or even the faulty/ malicious program that is responsible.

A. Threats to Cloud Accountability

1) $T_{2.2}$ – SLA violation: A. Haeberlen addresses the importance of accountability in cloud computing [47], where the loss of data control is problematic when something goes awry. For instance, the following problems may possibly arise: 1) The machines in the cloud can be misconfigured or defective and can consequently corrupt the customer's data or cause his computation to return incorrect results; 2) The cloud provider can accidentally allocate insufficient resources for the customer, an act which can degrade the performance of the customer's services and then violate the SLA; 3) An attacker can embed a bug into the customer's software in order to steal valuable data or to take over the customer's machines for spamming or DoS attacks; 4) The customer may not have access to his data either because the cloud loses it or simply because the data is unavailable at an inconvenient time. If something goes wrong, for example, data leaks to a competitor, or the computation returns incorrect results; it can be difficult for a customer and provider to determine which of them has caused the problem, and, in the absence of solid evidence, it is nearly impossible for them to hold each other responsible for the problem if a dispute arises.

2) $T_{2.3}$ – Dishonest MapReduce: MapReduce [53] is a parallel computing paradigm that is widely employed by major cloud providers (Google, Yahoo!, Facebook, etc.). MapReduce splits a large data set into multiple blocks, each of which are subsequently input into a single worker machine for processing. However, working machines may be misconfigured or malicious, as a result, the processing results returned by the cloud may be inaccurate. In addition, it is difficult for customers to verify the correctness of results other than by running the same task again locally. Dishonest MapReduce may be viewed as a concrete case of computation integrity problem, as we discussed in Section III (i.e., cloud integrity). The issue will be further addressed by accountability, because even after customers have verified the correctness of MapReduce output, there is still a necessity to identify the faulty machines or any other possible reasons that resulted in wrong answers.

3) $T_{2.4}$ – Hidden Identity of Adversaries: Due to privacy concerns, cloud providers should not disclose cloud customers' identity information. Anonymous access is employed to deal with this issue; although anonymity increases privacy, it also introduces security problems. Full anonymity requires that a customer's information must be completely hidden from absolutely anyone or anything else. In this case, malicious users can jeopardize the data integrity without being detected since it becomes easier to hide their identities.

4) $T_{4.2}$ – Inaccurate Billing of Resource Consumption:
The pay-as-you-go model enables customers to decide how to outsource their business based on their necessities as well as the financial situations. However, it is quite difficult for customers to verify the expenses of the resource consumption due to the black box and dynamic nature of cloud computing. From the cloud vendor's perspective, in order to achieve maximum profitability, the cloud providers choose to multiplex applications belonging to different customers to keep high utilization. The multiplexing may cause providers to incorrectly attribute resource consumption to customers or implicitly bear additional costs, therefore reducing their cost-effectiveness [18]. For example, I/O time and internal network bandwidth are not metered, even though each incurs non-trivial cost. Additionally, metering sharing effects, such as shared memory usage, is difficult.

B. Defense Strategies

1) $D_{2.2.1}$ – Accountability on Service Level Agreement (SLA): To deal with this dispute of an SLA violation, a primitive AUDIT (A, S, t1, t2) is proposed in [47] to allow the customers to check whether the cloud provider has fulfilled the SLA (denoted by A) for service S between time internal t1 and t2. AUDIT will return OK if no fault is detected; otherwise AUDIT will provide verifiable evidence to expose the responsible party. The author in [47] does not detail the design of AUDIT, instead the author provides a set of building blocks that may be contributive, including 1) tamper-evident logs that can record all the history actions of an application, 2) virtualization-based replays that can audit the actions of other applications by replaying their logs, 3) trusted time stamping that can be used to detect performance fault (i.e., latency or throughput cannot match the level in SLA), and 4) sampling that can provide probability guarantees and can improve efficiency of replay.

2) $D_{2.2.2}$ – Accountable Virtual Machine: Accountable Virtual Machine (AVM) [50] is follow-up work of [47]. The intent of AVM is to enable users to audit the software execution on remote machines. AVM is able to 1) detect faults, 2) identify faulty node, 3) provides verifiable evidence of a particular fault and point to the responsible party. AVM is applicable to cloud computing in which customers outsource their data and software on distrusted cloud servers. AVM allows cloud users to verify the correctness of their code in the cloud system. The approach is to wrap any running software in a virtual machine, which keeps a tamper-evident log [51] to record the entire execution of the software. If we assume there is a reference implementation, which defines the correct execution of the software, the cloud users have enough information to verify the software correctness by replaying the log file and comparing it with the reference copy. If there is an inconsistency, there will be mismatches detected. The log is tamper-evident, meaning that nobody may tamper with the log file without being detected. Once the integrity of the log file is ensured, the evidence obtained from it is trustworthy. The evidence is provable by any external party. One limitation of AVM is that it can only detect faults caused by network operations since it only logs network input/output messages.

3) $D_{2.2.3}$ – Collaborative Monitoring: A solution that is similar to AVM was developed in [48] by maintaining an external state machine whose job is to validate the correctness of the data and the execution of business logic in a multi-tenancy environment. The authors in [48] define the service endpoint as the interface through which the cloud services are delivered to its end users. It is assumed that the data may only be accessed through endpoints that are specified according to the SLA between the cloud provider and the users. The basic idea is to wrap each endpoint with an adapter that is able to capture the input/output of the endpoint and record all the operations performed through the endpoint. The log is subsequently sent to the external state machine for authentication purposes. To perform the correctness verification, the Merkle B-tree [49] is employed to authenticate the data that is stored in the cloud system. An update operation on the data will also update the MB-tree. A query operation is authenticated by a range query on the MB-tree. Once the correctness checking fails, the state machine will report problems and provide verifiable evidence based on the query result of the MB-tree.

4) $D_{2.3.1}$ – Accountable MapReduce: In [54], this problem has been addressed with SecureMR, which adopts full task duplication to double check the processing result. SecureMR requires that twice two different machines, which will double the total processing time, execute a task. Additionally, SecureMR suffers false positive when an identical faulty program processes the duplicated tasks.

Xiao et al. [66] propose to build an Accountable MapReduce to detect the malignant nodes. The basic idea is as follows: the cloud provider establishes a trust domain, which consists of multiple regular worker machines referred to as Auditors. An auditor makes use of the determinism of Map/Reduce functions in order to apply an Accountable Test (A-Test) for each task on each working machine. The AC picks up a task that has been completed by a machine M, then re-executes it and compares the output with M's. If an inconsistency shows up, then M is proved to be malicious. The A-Test will stop when all tasks are tested. A full duplication of an execution requires large computation costs. Instead of pursuing a 100% detection rate, the authors determined to provide probability guarantees in order to accelerate the A-test. At this moment, the general idea is to only re-execute a part of each task. By carefully selecting the parameters, high detection rate (e.g., 99%) may be achieved with low computation costs.

5) $D_{2.4.1}$ – Secure Provenance: Secure provenance is introduced with an aim to ensure that verifiable evidence might be provided to trace the real data owner and the records of data modification. Secure provenance is essential to improve data forensic and accountability in cloud systems. Lu et al.

[41] have proposed a secure provenance scheme based on bilinear paring techniques, first bringing provenance problems into cloud computing. Considering a file stored in cloud, when there is dispute on that file, the cloud can provide all provenance information with the ability to plot all versions of the file and the users that modified

it. With this information, a specific user identity can be tracked.

6) $D_{4.2.1}$ – Verifiable Resource Accounting: Sekar and Maniatis [18] have proposed verifiable resource accounting, which enables cloud customers to be assured that i) their applications indeed consumed the resources they were charged for and ii) the consumption was justified based on an agreed policy. The scheme in [18] considers three roles: the customer

C, the provider P, and the verifier V. First, C asks P to run task T; then, P generates a report R describing what resources P thinks that C consumes. C then sends the report R and some additional data to V who checks whether R is a valid consumption report. By implementing a trusted hardware layer with other existing technologies such as offloading monitoring, sampling, and snapshot, it can be ensured that a) the provider does not overcharge/undercharge customers and b) the provider correctly assigns the consumption of a resource to the principal responsible for using that resource.

7) Other Opinions: In order to practically include account-ability into cloud environment, Ko et al. [69] present the Cloud Accountability Life Cycle (CALC), describing the key phases to build cloud accountability. The CALC contains seven phases: 1) policy planning, 2) sense and trace, 3) logging, 4) safe-keeping of logs, 5) reporting and replaying, 6) auditing, and 7) optimizing and rectifying.

C. Summary and Open Issues

Accountability is a significant attribute of cloud computing because the computing
paradigm increases difficulty of holding an entity responsible for some action. Following a pay-as-you-go billing model, cloud vendor provides resources rented by customers who may host their web contents opening to public clients. Even a simple action (e.g., a web request) will involve multiple parties.
such as software bug, misconfiguration, and hard-ware failure, to help identify the event origin. Therefore, an accountable cloud will be a great step towards a trustworthy cloud.

There are a few open issues for future research:

- Accountable MapReduce enables accountability for MapReduce by offering verifiable evidence pointing to a responsible faulty/malicious node when either Map function or Reduce function is not faithfully performed. However, current solutions are based on replication, which has two drawbacks: a) replication is not efficient, and cloud vendors do not have much incentive to do so; sampling can improve efficiency but it reduce accuracy, which is even more important to customers; b) replication is effective only if cloud vendor follows semi-trust model, which may not be true in real world. Therefore, it is imperative to design new schemes that can achieve both efficiency and accuracy.

- To implement resource usage accountability, $D_{4.2.1}$ relies on cloud vendors to generate a consumption report that has the ability to be verified by a third party. Another idea is reducing the cloud provider’s support. There are two possible options [18]. The first option is resource prediction, which enables customers to predict workloads by mining the execution logs, and then compares the result with provider’s usage report. The primary concern of resource prediction is that it is not accurate enough. Another option is to enable multiple customers to detect violations collaboratively. However, it is hard to keep customers’ privacy if the two work together.

VI. Cloud Privacy

Privacy is yet another critical concern with regards to cloud computing due to the fact that customers’ data and business logic reside among distrusted cloud servers, which are owned and maintained by the cloud provider. Therefore, there are potential risks that the confidential data (e.g., financial data, health record) or personal information (e.g., personal profile) is disclosed to public or business competitors. Privacy has been an issue of the highest priority [132], [133], [134].

Throughout this text, we regard privacy-preservability as the core attribute of privacy. A few security attributes directly or indirectly influence privacy-preservability, including confidentiality, integrity, accountability, etc. Evidently, in order to keep private data from being disclosed, confidentiality becomes in-dispensable, and integrity ensures that data/computation is not corrupted, which somehow preserves privacy. Accountability, on the contrary, may undermine privacy due to the fact that the methods of achieving the two attributes usually conflict. More details will be given in this section.

A. Threats to Cloud Privacy

In some sense, privacy-preservability is a stricter form of confidentiality, due to the notion that they both prevent information leakage. Therefore, if cloud confidentiality is ever violated, privacy-preservability will also be violated. Similar to other security services

B. Defense Strategies

Chow et al. [59] have classified the privacy-preserving approaches into three categories, which are shown in Table II.

Gentry proposed Fully Homomorphic Encryption (FHE,) to preserve privacy in cloud computing [64]. FHE enables computation on encrypted data, which is stored in the distrusted servers of the cloud provider. Data may be processed without decryption. The cloud servers have little to no knowledge concerning the input data, the processing function, the result, and any intermediate result values. Therefore, the outsourced computation occurs 'under the covers' in a fully privacy-preserving way. FHE has become a powerful tool to enforce privacy preserving in cloud computing. However, all known FHE schemes are too inefficient for use in practice. While researchers are trying to reduce the complexity of FHE, it is worthwhile to consider alleviating the power of FHE to regain efficiency. Naehrig et al. has proposed somewhat homomorphic encryption [98], which only supports a number of homomorphic operations, which may be much faster and more compact than FHE.

Pearson et al. ([42] and [43]) propose privacy manager) that relies on obfuscation techniques. The privacy manager can provide obfuscation and de-obfuscation service to reduce the amount of sensitive information stored in the cloud. The main idea is to only store the encrypted form of clients' private data in the cloud end. The data process is directly performed on the encrypted data. One limitation is that cloud vendors may not be willing to implement additional services for privacy protection. Without provider's cooperation, this scheme will not work.

Squicciarini et al. [31] explores a novel privacy issue that is caused by data indexing. In order to tackle data indexing and to prevent information leakage, the researchers present a three-tier data protection architecture to offer different levels of privacy to cloud customers.

Itani et al. [44] presents a Privacy-as-a-Service so it may enable secure storage and computation of private data by lever-aging the tamper-proof capabilities of cryptographic coprocessors. Which, in turn, protect customer data from unauthorized access. Sadeghi et al. [58] argue that pure cryptographic solutions based on fully homomorphic and verifiable encryption suffer high latency for offering practical secure outsourcing of computation to a distrusted cloud service provider. They propose to combine a trusted hardware token ($D_{2.5.3}$) with Secure Function Evaluation (SFE) in order to compute arbitrary functions on data when it is still in encrypted form. The computation leaks no information and is verifiable. The focus of this work is to minimize the computation latency to enable efficient, secure outsourcing in cloud computing. A hardware token is tamper-proof against physical attacks. If the token is under the assumption of being trusty, the clients' data processing may be performed in the token that is attached to a distrusted cloud server. The property of a token can guarantee that the data computation is confidential as well as being verifiable. The solution presented in [58] only needs to deploy a tamper-proof token in the setup pre-processing phase.

TABLE II
APPROACHES OF PRIVACY ENFORCEMENT

Approach	Description	Example
Information centric security	Data objects have access-control policies with them.	A data outsourcing architecture combining cryptography and access control [9].
Trusted computing	The system will consistently behave in expected ways with hardware or software enforcement.	Trusted Cloud Computing Platform [29]; Hardware token [58]; Privacy-aaS [44].
Cryptographic protocols	Cryptographic techniques and tools are employed to preserve privacy.	Fully Homomorphic Encryption (FHE) [64] and its applications [58]

symmetric cryptographic operations are performed in the cloud, without requiring further interaction with the token.

1) Other Opinions: Cryptography is **NOT** all-purpose

Van Dijk et al. [63] argue that cryptography alone cannot provide complete solutions to all privacy issues in cloud computing, even with powerful tools like FHE. The authors formally define a class of privacy problems in terms of various application scenarios. It has been proved that when data is shared among customers, no cryptographic protocol can be implemented to offer privacy assurance. Let us define the following notations

[63]:

- S – the cloud, which is a highly resourced, monolithic entity.
- $C = C_1, C_2, ..., C_n$ – a set of customers/tenants of S.
- x_i – a static, private value belonging to C_i but stored in S.

The task of S is to run different applications/functions over $\{x_i\}$.
a) Class one: private single-client computing
These applications only process data x_i owned by a single client C_i. No other parties should be able to learn any information about the internal process. A typical example is a tax-preparation program taking as input financial data that belongs to a single client and should be hidden from both S and other clients. This class can be properly solved by an FHE that meets all the secure requirements.

b) Class two: private multi-client computing

These applications operate on data set $\{x_i\}$ owned by multiple clients $\{C_i\}$. Since there are more clients involved, data privacy among clients is preserved in a more complicated way. There are access-control policies that must be followed when processing data. A real world application is a social networking system, in which x_i is the personal profile of a client C_i; C_i is able to specify which friends can view what portions/functions of her data (i.e., gives an access control policy). It has been proved that private multi-client computing is unachievable using cryptography.

c) Class three: stateful private multi-client computing

This class is a restricted version of class two. The difference is that the access-control policy on a client's data is stateful, meaning that it depends on the application execution history. This class is not discussed thoroughly in the paper [63], but the authors do believe it has an important position in cloud applications.
2) Other Opinions: Privacy Preserving Frameworks

Lin et al. presented a general data protection framework [56] in order to address the privacy challenges in cloud service pro-vision. The framework consists of three key building blocks:

1) a policy ranking strategy to help cloud customers identify a cloud provider that best fits their privacy requirements; 2) an automatic policy generation mechanism that integrates the policies and requirements from both participants and produces a specific policy as agreed by them; 3) policy enforcement that ensures the policy will be fulfilled. A straightforward path to obtain policy ranking is comparing the customer requirement with the policies of multiple service providers and subsequently picking the one with the highest rank. The comparison may happen on the client's side, the cloud provider side, or through a broker. Policy algebra can be employed to carry out the policy generation. Each policy should be first formalized and then integrated with fine-grained policy algebra [57]. Pearson [28] suggests that privacy should be taken into account from the outset and should be considered in every phase of cloud service design.

C. Open Issues
Regarding cloud privacy, there are some open issues to be studied in future researches:

- The authors think that accountability and privacy may conflict with each other. The enforcement of account-ability will violate privacy in some degree, and extreme privacy protection (e.g., full anonymity to hide users' identity) will make accountability more challenging. An extreme example, a shared file, accessed by multiple users who, may hide their identities due to anonymity for the purpose of privacy protection. However, malicious users are tracked with difficultly because of the anonymous access. From the viewpoint of accountability, general approaches include information logging, replay, tracing [22], etc. These operations may not be completed without revealing some private information (e.g., account name, IP address). We must seek a trade-off in which the requirement of one attribute can be met while simultaneously maintaining some degree of the other attribute.

- The assessment of attributes is another important issue since it provides a quantitative way to evaluate them. The goal is to determine how secure a cloud is or how much privacy can be offered. The meaning is twofold: 1) it will be helpful to compare different security approaches; for example, to achieve 100% privacy, scheme A costs 100; scheme B can achieve 99% accountability with cost of 10. Apparently, scheme B is more practically efficient, although it sacrifices one percent of accountability. Without an assessment, it is difficult to compare two strategies quantitatively. 2) The quantitative clauses of the security/privacy requirements can be drafted into the Service Level Agreements (SLAs).

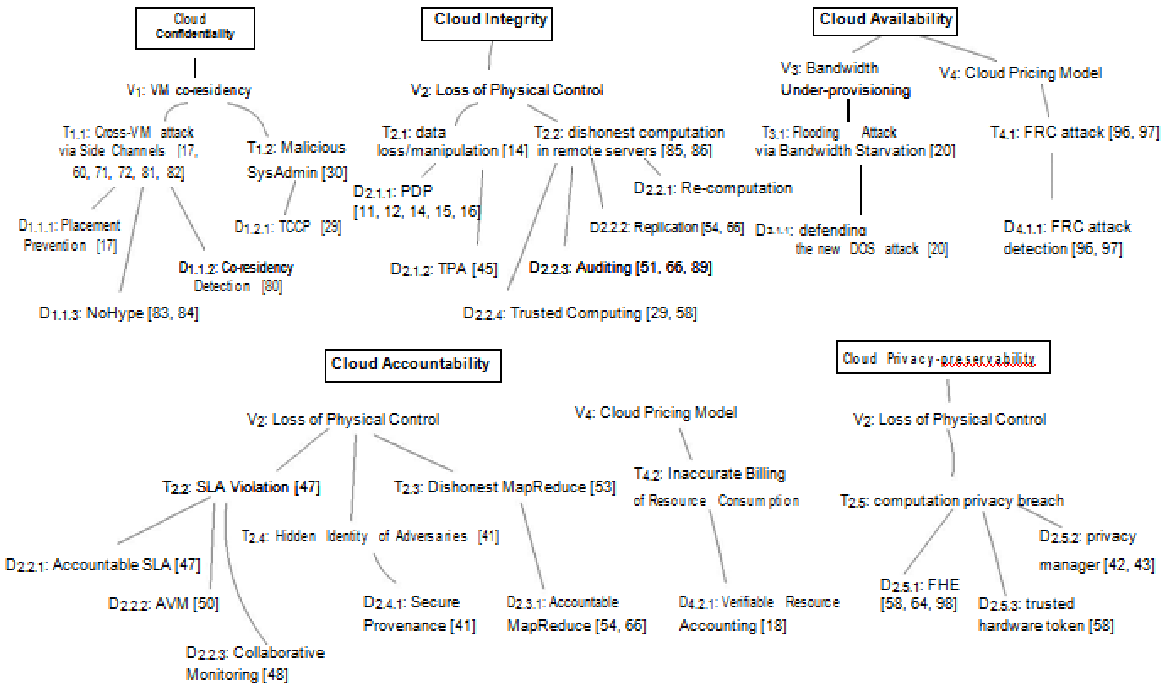

Cloud Confidentiality
V1: VM co-residency
T1.1: Cross-VM attack via Side Channels [17, 60, 71, 72, 81, 82]
T1.2: Malicious SysAdmin [30]
D1.1.1: Placement Prevention [17]
D1.2.1: TCCP [29]
D1.1.2: Co-residency Detection [80]
D1.1.3: NoHype [83, 84]
Cloud Integrity
V2: Loss of Physical Control
T2.1: data loss/manipulation [14]
T2.2: dishonest computation in remote servers [85, 86]
D2.1.1: PDP [11, 12, 14, 15, 16]
D2.2.1: Re-computation
D2.2.2: Replication [54, 66]
D2.1.2: TPA [45]
D2.2.3: Auditing [51, 66, 89]
D2.2.4: Trusted Computing [29, 58]
Cloud Availability
V3: Bandwidth Under-provisioning
V4: Cloud Pricing Model
T3.1: Flooding Attack via Bandwidth Starvation [20]
T4.1: FRC attack [96, 97]
D3.1.1: defending the new DOS attack [20]
D4.1.1: FRC attack detection [96, 97]
Cloud Accountability
V2: Loss of Physical Control
V4: Cloud Pricing Model
T2.2: SLA Violation [47]
T2.3: Dishonest MapReduce [53]
T4.2: Inaccurate Billing of Resource Consumption
T2.4: Hidden Identity of Adversaries [41]
D2.2.1: Accountable SLA [47]
D2.2.2: AVM [50]
D2.2.3: Collaborative Monitoring [48]
D2.4.1: Secure Provenance [41]
D2.3.1: Accountable MapReduce [54, 66]
D4.2.1: Verifiable Resource Accounting [18]
Cloud Privacy-preservability
V2: Loss of Physical Control
T2.5: computation privacy breach
D2.5.2: privacy manager [42, 43]
D2.5.1: FHE [58, 64, 98]
D2.5.3: trusted hardware token [58]

Fig. 5. A Summary of Research Advances in Cloud Security and Privacy

VII. Conclusions

Throughout this paper, the authors have systematically studied the security and privacy issues in cloud computing based on an attribute-driven methodology, shown in Fig. 5. We have identified the most representative security/privacy attributes (e.g., confidentiality, integrity, availability, account-ability, and privacy-preservability), as well as discussing the vulnerabilities, which may be exploited by adversaries in order to perform various attacks. Defense strategies and suggestions were discussed as well. We believe this review will help shape the future research directions in the areas of cloud security and privacy.

References

[1]. **Towards Secure and Dependable Storage Services in Cloud Computing** Cong Wang, Student Member, IEEE, Qian Wang, Student Member, IEEE, Kui Ren, Member, IEEE, Ning Cao, Student Member, IEEE, and Wenjing Lou, Senior Member, IEEE

[2]. **What's New About Cloud Computing Security** Yanpei Chen, Vern Paxson, Randy H. Katz CS Division, EECS Dept. UC Berkeley {ychen2, vern, randy}@eecs.berkeley.edu

[3]. **Securing Data Storage in Cloud Computing** Hyun-Suk Yu

[4]. **Identity-Based Authentication for Cloud Computing** Hongwei Li1, Yuanshun Dai1,2, Ling Tian1, and Haomiao Yang1

[5]. **A Study on Secure Data Storage Strategy in Cloud Computing** Danwei Chen,Yanjun He First AuthorCollege of Computer Technology, Nanjing University of Posts and Telecommunications, chendw@njupt.edu.cn

[6]. **Cloud Computing Research and Security Issues** Jianfeng Yang College of Computer Science and Technology Sichuan University

[7]. **Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities** Rajkumar Buyya1,2, Chee Shin Yeo1, and Srikumar Venugopal

[8]. **Mobile Cloud Computing: A Comparison of Application Models** Dejan Kovachev, Yiwei Cao and Ralf Klamma Information Systems & Database Technologies RWTH Aachen University

[9]. **9. Efficient and Secure Data Storage Operations for Mobile Cloud Computing** Zhibin Zhou and Dijiang Huang {zhibin.zhou,dijiang}@asu.edu Arizona State University