Malla Reddy College Engineering (Autonomous)

Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad, Telangana-500100 www.mrec.ac.in

Department of Information Technology

II B. TECH II SEM (A.Y.2018-19)

Lecture Notes

On

80535 - Cloud Computing

Module III – cloud computing Architecture over Virtualized Data centers :- (1)

Data Center Design and interconnection networks:-

→ A data center is often built with a large number of servers through a huge interconnection network.

→ There are large-scale data centers and small modular data centers that can be housed in a 40-ft truck container.

Ware-house-Scale Data-Center Design:-

→ Dennis Gannon claims: "The cloud is built on massive data centers". The data centers are built economics of scale – meaning lower unit cost for larger data centers: Datahouse it has 400,000 to 1 million servers. cost to operate 400-server facility $13/mbps; storage $0.4/GB. The A small data center could have 1,000 servers. The larger the data center, the lower the operational cost. Microsoft has 100 data centers, large or small across globe.

Data-Center construction Requirements:-

→ Most data centers are built with commercially available components. An off-the-shelf server consists of a number of processor sockets, each with multicore CPU and its internal cache hierarchy, local shared and coherent DRAM, and a number of directly attached disk drives.

→ The DRAM and disk resources within the rack are accessible through first-level rack switches and all resources in all racks are

→ are accessible via a cluster level switch. Consider a data center built with 2000 servers, each with 8GB of DRAM and four 1TB disk drives. Each group of 40 servers is connected through a 1 Gbps link to a rack-level switch that has an additional eight 1 Gbps ports used for connecting the rack to the cluster-level switch.

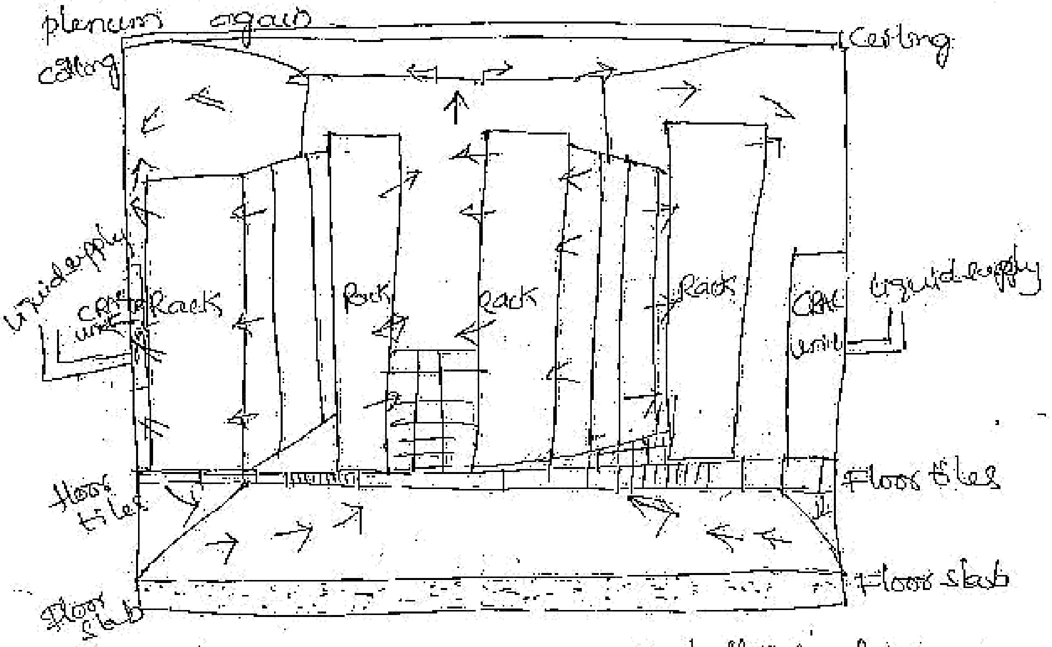
→ It was estimate that the bandwidth available from local disk is 200 MB/s, whereas the bandwidth from off-rack disks is 25 MB/s via shared rack uplink. The ~~total~~ disk storage in the cluster is almost 10 million times larger than local DRAM.

Cooling System of a Data-Center Room:-

→ The data center room has raised floors for hiding cables, power lines, and cooling supplies. The room it has a steel grid resting on stanchions about 2-4 ft above the concrete floor.

→ The under-floor area is often used to route power cables to racks, but its primary use is to distribute cool air to the server rack. The CRAC (computer room air conditioning) unit pressurizes the raised floor plenum by blowing cold air into the plenum.

→ The cold air escapes from the plenum through perforated tiles that are placed in front of server racks. Racks are arranged in long aisles that alternate between cold aisles and hot aisles to avoid mixing hot and cold air. The hot air produced by (3) the servers circulates back to the intakes of the incoming coolant ~~us~~ at CRAC units that cool it and then exhaust the cool air into the raised floor plenum again.



A cooling system in a raised floor data center

Data Center interconnection Networks :-

→ A critical core design of a data center is the interconnection network among all servers in the data center cluster.

→ This n/w design must meet five special requirements: 1) low latency, high bandwidth, low cost, message-passing interface (MPI) communi-cation support, and fault tolerance. The design of an inter-server n/w must satisfy both point-to-point and collective communication patterns among all server nodes.

Specific design considerations are:

1) Application Traffic support :- The n/w topology should support all MPI communication patterns. Both point to point and collective MPI communications must be supported. The network should have high bandwidth to meet this requirement.

2) Network Expandability :-

→ The interconnection network should be expandable.

→ With thousands or even hundreds of thousands of server nodes, the cluster n/w interconnection should be restructured while being allowed to expand one more servers are added to the data center.

→ Hardware should be designed to support load balancing and data movement among the servers.

→ None of the links should be down.

Fault tolerance and Graceful degradation :-

→ It should provide some mechanism to tolerate link or switch failures.

→ fault tolerance of server is achieved by replicating data and computing among redundant servers.

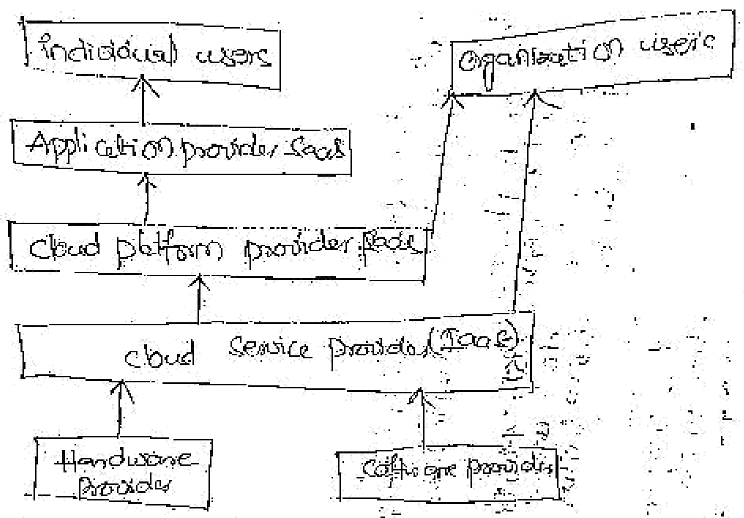
→ Both h/w and s/w should be handled or cope with network failures.

→ There should not critical paths which may become a single point of failures pulls down the entire system.

Public cloud platforms: GAE, AWS, AND AZURE

Public clouds and service offerings:

→ Cloud services are demanded by computing and IT administrators, software vendors and end users.



Individual users
Organization users
Application provider SaaS
Cloud platform provider PaaS
cloud service provider (IaaS)
Hardware provider
Software providers

Roles of individual and organizational users and their interaction with cloud providers under various cloud service models.

→ At the top level, individual users and organizational users demand very different services. The application providers at the SaaS level serve mainly individual users.

→ Most business organizations are served by IaaS and PaaS providers.

→ The IaaS service provide compute, storage and communication resources to both applications and organizational users. The cloud environment is defined by PaaS (or) platform providers.

→ The provider service charges are often much lower than the cost for users to replace their obsolete servers frequently.

Five major cloud platforms and their service offerings.

Model	IBM	Amazon	Google	Microsoft	Salesforce
PaaS	Blue cloud, WCA, RC2	-	App Engine (GAE)	Windows Azure	Force.com
IaaS	Ensembles	AWS	-	windows Azure	-
SaaS	Lotus Live	-	Gmail, docs	.NET service, Dynamic CRM	online CRM, Gifting
Virtualization	-	OS and Xen	Application container	OS level / HyperV-V	~~online CRM, Gifting~~
Service offerings	SOA, B2, TSAM, RAD web2.0	EC2, S3, SQS, simple DB.	GFS, Chubby, BigTable, MapReduce	Live, SQL Hotmail	Apex, visual force, record security.
Security features	Web sphere2 and PowerVM tuned for protection	PKI, VPN, EBS to recover from failure	chubby locks for security enforcement	Replicated data rule based access control	Admin/record security, uses metadata API
user interface	—	EC2 command line tools	web based admin console	windows Azure portal	—
web programming support	Yes AMI	Yes.	Yes Python	Yes .NET framework	Yes

→ Amazon pioneered the IaaS business in supplying e-commerce and cloud applications by millions of customers simultaneously.

→ The elasticity in the Amazon cloud comes from the flexibility provided by the hardware and software services.

→ S3 provides unlimited online storage space. EC2 provides an environment for running virtual servers on Demand. Both EC2 and S3 are supported in the AWS platform.

→ Microsoft offers the Azure platform for cloud applications. It has also supported the .NET service, dynamic CRM, Hotmail, and SQL applications.

→ Salesforce.com offers extensive SaaS applications for online CRM applications using its Force.com platforms.

→ All IaaS, PaaS and SaaS models allow users to access users to access services over the Internet, relying entirely on the infrastructure of the cloud service providers.

→ These models are offered based on [illegible] between the providers and their users.

[illegible]

→ Service level aggrements (SLA) are common in network services, it is ~~difficult~~ ~~to~~ account for the QoS Characteristics of network services.

→ In a broder sense, the service level aggrements (SLAs) for cloud computing address service availability, data integrity, privacy, and security protection. Blank spaces in the ~~table~~ refer to unknown or underdeveloped features.

Google App Engine (GAE):-

→ Google has the world's largest search engine facilities.

→ Google has hundreads of data centers and has installed more than 460,000 Servers worldwide.

→ Data items are stored in text, images, and vedio and are replicated to tolerate faults on failures.

→ Google's AppEngine (GAE) offers a PaaS platform supporting various cloud and web applications.

1. Google cloud Infrastructure:-

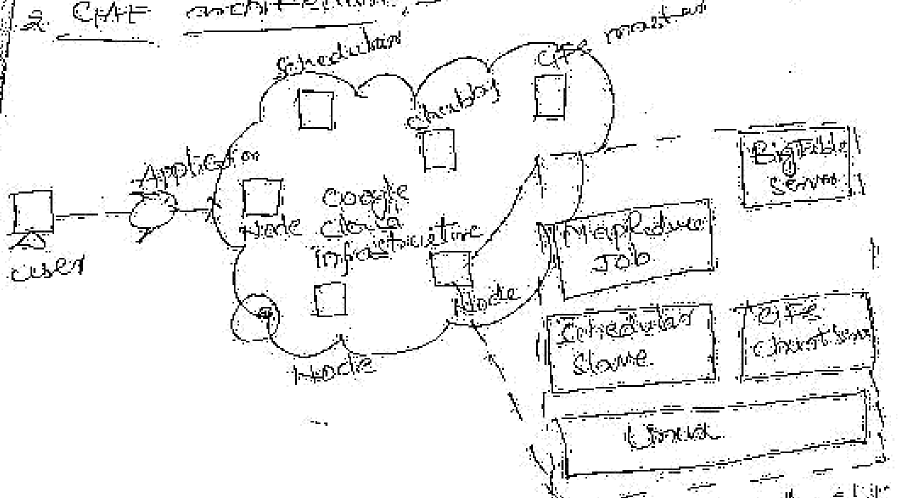
→ Google has pioneered development by leveraging the large number of data centers it operates. For examples, Gmail, google docs, and google Earth, other application are google pioneed cloud services.

→ Notable technology achievements of google include Google file System (GFS), MapReduce, BigTable, and Chubby.

→ In 2008, google announced the GAE web application platform which is becoming a common platform for many small cloud service providers. This platform specializes in supporting of ~~[illegible]~~ scalable (elastic) web applications

→ GAE enables users to run their applications on a large number of data centers associated with Google's search engine operations.

2. GAE architecture:-



Google cloud platform and major building blocks

→ GFS is used to store large amounts of data. MapReduce is for use in application program development

→ chubby is used for distributed application lock services. BigTable offers a storage service for accessing structured data.

→ users can interact with google application via the web interface provided by each application.

→ third-party application providers can use GAE to build cloud applications for providing services

→ The applications all run in data centers under tight management by google engineers.

→ A typical cluster configuration can run the Google File System, MapReduce jobs, and BigTable servers for structure data. Extra services such as chubby for distributed locks can also run in the clusters.

→ GAE runs the user program on Google's infrastructure. And outside people cannot use the Google infrastructure to build their own service because its fundamental service program is private.

→ Application developers do not need worry about the maintenance of servers; As it is a platform running third-party programs.

→ GAE supports python and Java programming environments.

3. Functional Modules of GAE:-

The GAE platform comprises the following major components.

a) The datastore offers object-oriented, distributed, structured and storage services based on BigTable techniques.

b) the application runtime environment offers a platform for scalable web programming and execution. It supports two development languages python and Java.

c) The software development Kit (SDK) is used for local application development. The SDK allows user to execute test runs of local applications and upload application code.

d) The administration console is used for easy management of user application development cycles, instead of for physical resource management.

e) The GAE web service infrastructure provides special interfaces to guarantee flexible use and management of storage and network resources by GAE

Google offers essentially free GAE services to all Gmail account owners. You can register for a GAE account or use your Gmail account name to signup for the service. The service is free within a quota. If you exceed the quota, the page instructs you to how to pay for the service.

→ The platform doesnot provide any IaaS services, unlike Amazon, which offers IaaS and PaaS. This allows developers to develop the applications using programming languages and the software tools supported by the provider.

4) GAE applications :-

→ Well-Known GAE applications, include the Google Search Engine, Google Docs, Google Earth, and Gmail.

→ These applications can support large numbers of users simultaneously.

→ The applications are all run in the Google data centers.

→ GAE supports many web applications. One is storage service to store application-specific data in the google infrastructure.

→ GAE also supports Google-specific services, such as Gmail account service (which is a login service, that is, applications can use the Gmail account directly.

→ Web applications built on top of GAE can use the APIs authenticating users and sending e-mail using Google accounts.

→ Resource Management and Energy - Efficiency:-

→ One important challenge faced by providers of cloud computing services is the efficient management of virtualized resource pools. Physical resources such as CPU cores, disk space, and network bandwidth must be sliced and shared among virtual machines running potentially heterogeneous workloads.

→ Data centers consumes large amounts of electricity.

→ According to a data published by HP, 100 server racks can consume 1.3 MW of power and another 1.3 MW are required by the cooling system - -

→ To optimize application performance, dynamic resource management can also improve utilization and consequently minimize energy consumption in data centers.

Nutshell: briefly. Module-I cloud computing

→ Many practitioners in the commercial and academic spheres have attempted to define exactly what "cloud computing" is and what unique characteristics it presents.

Buyya et al. have defined it as follows: "Cloud is a parallel and distributed computing system consisting of a collection of interconnected and virtualised computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements (SLA) established through negotiation between the service provider and consumers".

→ The National Institute of Standards and Technology (NIST) characteristics cloud computing as network access to a pay-per-use model for enabling available, convenient, on-demand network access to a shared pool of configurable computing resources (eg. networks, servers, storage, applications, services).

Armbrust et al. define cloud as the "data center hardware and software that provide services.".

→ In addition to raw computing and storage, cloud computing providers usually offer a broad range of software services.

→ They also include APIs and development tools that allow developers to build seamlessly scalable applications upon their services.

→ The ultimate goal is allowing customers to run their everyday IT infrastructure "in the cloud".

→ Which a cloud should have common characteristics
(i) pay-per-use (no ongoing commitment, utility prices).
(ii) elastic capacity and illusion of infinite resources.
(iii) self-service interface.
(iv) resources that are abstracted or virtualised.

→ A report from the University of California Berkeley summarized the key characteristics of cloud computing as:
1) the illusion of infinite computing resources;
(2) the elimination of an up-front commitment by cloud users;
(3) the ability to pay for use ... as needed.

Armbrust et al. propose definitions for public cloud as "cloud made available in a pay-as-you-go manner to the general public."

→ and Private cloud as "internal data center of a business or other organization, not made available to the general public."

→ This allows users to interact with the local data center while experiencing the same advantages of public clouds, privileged access to virtual servers, and per-usage metering and billing.

→ A community cloud is "shared by several organizations and supports a specific community that has shared concerns.

→ A hybrid cloud its a combination of public and private clouds. It takes shape when a private cloud supplemented with computing capacity from public clouds. The approach of temporarily renting capacity to handle spikes in load is known as "cloud-bursting".

System models for Distributed and cloud computing:- 4

→ Distributed and cloud computing systems are built over a large number of autonomous computer nodes.

→ These node machines are interconnected by SANs, LANs, or WANs in a hierarchical maner.

→ With today's networking technology, a few LAN switches can easily connect hundreds of machines as a working cluster.

→ A WAN can connect many local system clusters to form a very large cluster of clusters.

→ In this sense, one can build a massive system with millions of computers connected to edge networks.

→ The massive systems are classified into four groups:
1> clusters 2> P2P networks 3> computing grids 4> Internet clouds over huge data centers.

→ clusters of cooperative computers

→ A computing cluster consists of interconnected stand-alone computers which work cooperatively as a single integrated computing resource.

→ In the past, clustered computers systems have demonstrated impressive results in handling heavy workloads with large data sets.

Peer-to-peer (P2P) networks:-

→ In a P2P system, every node acts as both a client and a server, Providing part of the system resources.

→ Peer machines are simply client computers connected to the Internet.

→ Peers act autonomously to join or leave the network. No central coordination (or) central database is needed.

→ No peer machine has a global view of the entire P2P system. The system is self-organizing with distributed control.

→ Unlike a cluster or grid, a P2P network does not use dedicated interconnection network.

P2P Networks are classified into different groups:

Major categories of P2P Network families:

Distributed file sharing: content distribution of mp3 music, video, etc.

Eg: Gnutella, Napster, BitTorrent.

Collaboration P2P networks: skype chatting, instant messaging, gaming etc.

Distributed P2P computing: specific application computing such as SETI@ home provides 25 Tflops of distributed computing power over 3 million Internet host machines.

8. computing grids:-

→ Grids are heterogeneous clusters interconnected by high-speed networks.

→ They have centralized control, are server-oriented with authenticated security. They are suited to distribute

→ In case of failures, the n/w structure should degrade gracefully amid limited node failures.

→ The n/w structure is often divided into two layers. The lower layer is close to the end servers, and the upper layer establishes the backbone connection among the server groups or sub-clusters.

Switch centric Data-center design:-

Two approaches to building data center scale n/ws. One is switch centric and the other one is server-centric.

→ In switch centric network, switches are used to connect the server nodes.

→ In server centric modifys the operating system running on the servers.

Modular Data center:-

→ A modern data center is structured as shipyard of server clusters housed in truck-towed containers.

→ Inside the container, hundreds of blade servers are housed in racks surrounding the container walls. Data centers usually are built at a site where leases and utilities for electricity are cheaper, and cooling is more efficient.

→ An array of fans forces the heated air generated by the server racks to go through a heat exchanger, which cools the air for the next rack on a continuous loop.

Container Data-center construction:-

→ The data-center module is housed in a truck-movable container. The modular container design includes the n/w, computer, storage, and cooling gear.

→ One needs to increase cooling efficiency by varying the water and airflow with better airflow management. The construction of a container-based data center may start with one system (one server), then move to rack design, and finally to a container system. Building a rack of 40 servers may take a half day. Expanding this to a whole container system with multiple racks for 1,000 servers requires the layout of the floor space with power, networking, cooling and complete testing.

→ The container must be the designed to be a weatherproof and easy to transport.

Interconnection of modular data center:-

→ container based data-center modules are meant for construction of even larger data centers using a farm of container modules.

→ 1. A server-centric design of data center module developed by Guo, et al. for interconnecting modular data centers.

→ The servers are represented by circles, and switches by rectangles, the Bcube provides a layered structure

supercomputing.
Eg:- TeraGrid.

→ Like an electric utility power, a computing grid offers an infrastructure that couples computers, software/middleware, people, and sensors together.

→ The grid is constructed across LANs, WANs, or Internet backbones at a regional, national, or global scale.

→ The computers used in a grid include servers, clusters, and supercomputers. PCs, laptops, and mobile devices can be used to access a grid system.

4. Internet clouds:-

→ cloud computing has been defined differently by many users and designers. For example, IBM, a major player in cloud computing, has defined it follows: "A cloud is a pool of virtualized computer resources. A cloud can host a variety of different workloads, including batch-style backend jobs and interactive and user-facing applications."

→ workloads can be deployed and scaled out quickly through rapid provisioning of VMs. Virtualization of server resources has enabled cost effectiveness and allowed cloud systems to leverage low costs to benefit both users and providers.

→ cloud systems should be able to monitor resource usage in real time to enable rebalancing of allocations when needed.

→ cloud computing applies a virtualized platform with elastic resources on demand by provisioning hardware, software, and datasets dynamically.

→ Desktop computing is moved to a service-oriented platform using server clusters and huge databases at datacenters.

Data Center Management Issues:-

- The Basic requirements for managing the resources of a data center are.

Making a common users happy:- The data center should be provide quality service.

Controlled information flow:- Information flow should be streamlined sustained services and high availability are the primary goals.

Multiuser manageability:- The system must be managed to support all functions of a data center, including traffic flow, database updating and server maintenance.

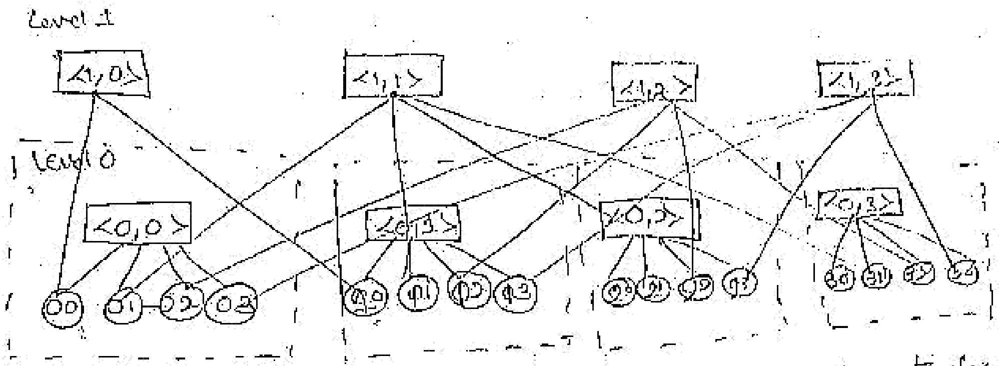
Scalability to prepare for database growth:- The storage, processing, I/O, power and cooling subsystems should be available.

Reliability in virtualized infrastructure:- Failover, fault-tolerance, and vm live migration should be integrated to enable recovery of critical applications from failures or disasters.

Low cost to both users and providers:- The cost to users and providers of the cloud system build over the data centers should be reduced, including all operational costs.

Security enforcement and data protection:- Data privacy and security defence mechanisms must be deployed to protect the data center against n/w attacks and system interrupts to maintain data integrity from n/w attacks.

Green information technology:- Saving power consumption and upgrading energy efficiency are in high demand when designing and operating current and future data centers.



Bcube, a high performance server centric network for building modular data center.

→ The bottom layer contains all the server nodes and they form Level 0. Level 1 switches form the top layer of Bcube. Bcub is a recursively constructed structure. The Bcube provides multiple paths between any two nodes.

2. Inter-Module connection Networks:-

→ Wu, et al have proposed a network topology for intercontainer connection using the aforementioned Bcub network as building blocks. Bcub is commonly used inside a server container. The proposed network was named

→ MDcub (for modularized Datacenter cube). This network connects multiple Bcube containers by using high-speed switches in the Bcube.

of their capacity,

→ In fact, mainframes had to operate at very high utilization rates simply because they were very expensive and costs should be justified by efficient usage.

→ The mainframe era collapsed with the advent of fast and inexpensive microprocessors and IT data centers moved to collections of commodity servers.

SOA, web services, web 2.0, and Mashups :-

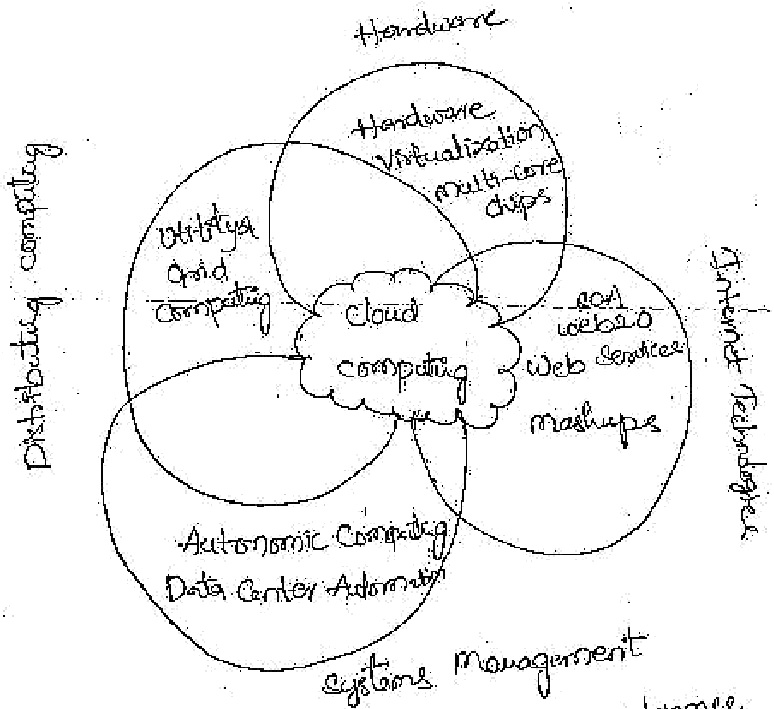
→ The emergence of Web services (WS) open standards has significantly contributed to advances in the domain of s/w integration. WS standards have been created on top of existing ubiquitous technologies such as HTTP and XML, thus providing common mechanism for delivering service, making them ideal for implementing a service oriented architecture (SOA).

→ The purpose of a SOA is to address requirements of loosely coupled, standards based, and protocol-independent distributed computing. In a SOA, software resources are packaged as "services", which are well defined, self-contained modules that provide standard business functionality and are independent of the state or context of other services.

→ The concept of giving services initially focused on the enterprise web, but gained space in the consumer realm as well, especially with the advent of web 2.0.

Roots of cloud computing:-

We can track the roots of clouds computing by observing the advancement of several technologies, especially in hardware (virtualization, multi-core chips), Internet technologies (web services, service-oriented architectures, web 2.0), distributed computing (clusters, grids), and systems management (autonomic computing, data center automation).



Hardware
Hardware Virtualization multi-core chips
Utility & Grid computing
Distributing computing
Cloud computing
SOA web 2.0 Web services mashups
Internet Technologies
Autonomic Computing Data Center Automation
systems Management

fig: Convergence of various advance leading to the advent of cloud computing.

→ From mainframes to clouds:-

In 1970s, companies who offered common data processing tasks, such as payroll automation, operated time-shared mainframes as utilities, which could serve dozens of applications and often operated close to 100

Architectural design of compute and storage clouds

A-Generic cloud Architecture design:-

→ An internet cloud is envisioned as a public cluster of servers provisioned on demand to perform collective web services or distributed applications using data center resources.

cloud platform design goals:- scalability, virtualization, efficiency, and reliability are four major design goals of a cloud computing platform.

→ cloud management receives the user request, finds the correct resources, and then calls the provisioning services which invoke the resources in the cloud.

→ System scalability can benefit from cluster architecture if one service takes a lot of power, storage capacity, or network traffic, it is simple to add more servers and bandwidth.

→ System reliability can benefit from this architecture. data can be put into multiple locations. For example: user e-mail can be put in three disks which expand to different geographically separate data centers.

→ If one of the data center crashes, user data is still accessible. the scale of cloud architecture can be increased by adding more servers.

Enabling technologies for clouds:-

clouds are enabled by progress in hardware, software, and networking technologies.

cloud Enabling Technologies in Hardware, software and Networking.

Technology		Requirements and Benefits
Fast platform deployment	-	fast, efficient, and flexible deployment of cloud resources to provide dynamic computing environment to users.
Virtual clusters on demand	-	Virtualized cluster of VMs provisioned to satisfy user demand and virtual cluster reconfigured as workload changes.
Multitenant techniques	-	SaaS for distributing software to a large number of users for their simultaneous use and resource sharing if so desired.
Massive data Processing	-	Internet search and web services which often require massive data, processing, especially to support personalized services.
web-scale communication	-	Support for e-commerce, distance education, social networking, digital entertainment applications.
Distributed storage	-	Large scale storage of personal records and public archive information which demands distributed storage over the clouds.
Licensing and billing services	-	License management and billing services which greatly benefit all types of cloud services in utility computing.

→ In the consumer web, information and services may be programmatically aggregated, acting as building blocks of complex compositions, called Service mashups.

" Cloud Computing Technologies = ① Virtualization ② SOA ③ Grid Computing ④ Utility Computing

Grid computing:—

→ Grid computing enables aggregation of distributed resources and transparently access to them.

→ Most production grids such as TeraGrid and EGEE seek to share compute and storage resources distributed across different administrative domains, with their main focus being speeding up a broad range of scientific applications, such as climate modeling, drug design, and protein analysis.

→ A key aspect of the grid vision realization has been building standard web services-based protocols that allow distributed resources to be rediscovered, accessed, allocated, monitored, accounted for, and billed for, etc. and in general managed as a single virtual system. The Open Grid Services Architecture (OGSA) addresses this need for standardization by defining a set of core capabilities and behaviors that address key concerns in grid systems.

Utility computing:—

In utility computing environments, users assign a "utility" value to their jobs, where utility is a fixed or time-varying valuation that captures various QoS contraints (deadline, importance, satisfaction).

→ The valuation is the amount they are willing to pay a service provider to satisfy their demands. The service providers then attempt to maximize their own utility, where said utility may directly correlate with their profit. providers can choose to prioritize high yield (i.e., profit per unit of resource) user jobs, leading to a scenario where shared systems are viewed as a marketplace, where users compete for resources based on the perceived utility or value of their jobs.

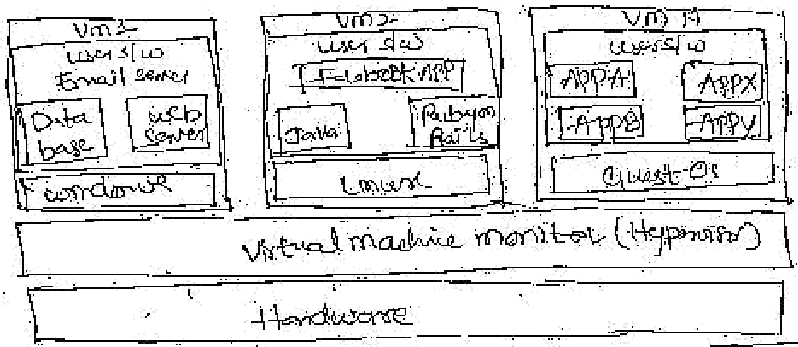
Hardware virtualization :-

→ cloud computing services are usually backed by large-scale data centers composed of thousands of computers. Such data centers are built to serve many users and host many disparate applications. For this purpose, hardware virtualization can be considered as a perfect fit to overcome most operational issues of data center building and maintenance.

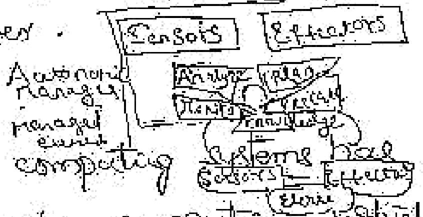
→ Hardware virtualization allows running multiple operating systems and software stacks on a single physical platform. A software layer, the virtual machine monitor (VMM), also called a hypervisor, mediates access to the physical hardware presenting to each guest operating system a

Virtual machine (VM), which is a set of virtual platform interfaces.

→ The advent of several innovative technologies multicore chips, para virtualization, hardware assisted virtualization, and live migration of VMs – has contributed to an increasing adoption of virtualization on server systems.



A h/w virtualized server.



Autonomic computing:–

→ The increasing complexity of computing systems has motivated research on autonomic computing, which seeks to improve systems by decreasing human involvement in their operations. In other words, systems should manage themselves, with high-level guidance from humans.

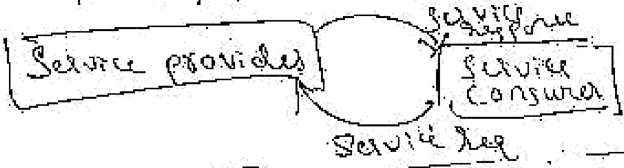
→ Autonomic, or self managing, systems rely on monitoring probes and gauges (sensors), on an adapting engine (autonomic manager) for computing optimizations based on monitoring data, and on effectors to carryout changes on the system. (18)

→ IBM's Autonomic computing initiative has contributed to define the four properties of autonomic systems: self configuration, self-optimization, self healing, and self-protection.

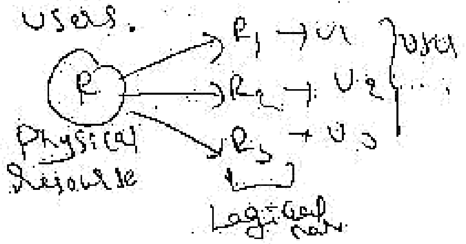
CC - Technology -

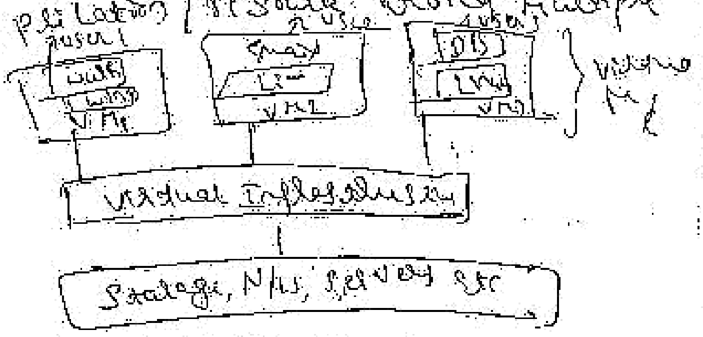
(i) SOA → Service are provided to form application through internet

Adv = Service Reusability, Easy maintenance, platform independent, Reliability, Scalab.



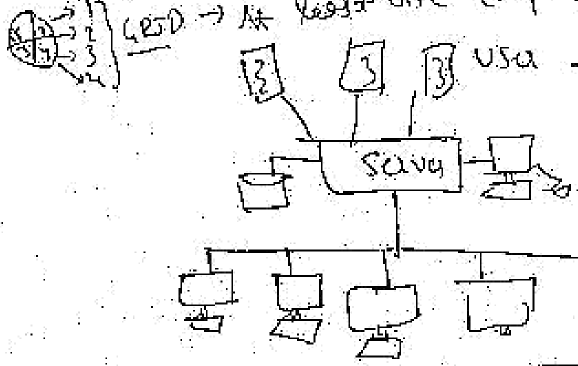
Virtualization: Allow to share single physical instance of an application / resource among multiple users.





Multitenant Architecture

(iii) GRID computing ⇒ Distributed computing in which a group of computers from multiple locations are connected with each other
- Computer Resources can be heterogeneous & can be located anywhere,
GRID → At least one computer in a group working as a server
user → Special grid computing S/W is used



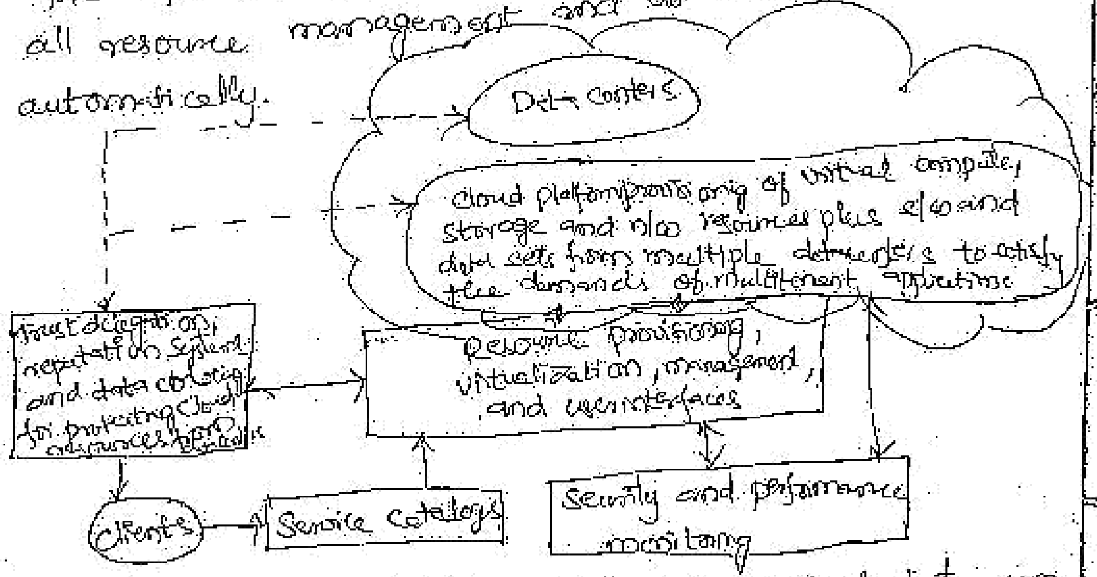
(iv) Utility computing
↳ pay per use model
↳ Resources are used as utility, with watch

A Generic cloud architecture:-

→ The internet cloud is envisioned as a massive cluster of servers. These servers are provisioned on demand to perform collective web services or distributed applications using data-center resources.

→ Servers in cloud can be physical machines or VM machines. User interfaces are applied to request services and provisioning tool carves out the cloud system to deliver the requested service.

→ The software infrastructure of cloud must handle all resource management and do most of maintenance automatically.



Data centers
cloud platform provisioning of virtual compute, storage and n/w resources plus e/o and data sets from multiple datacenters to satisfy the demands of multitenant applications
Trust delegation, reputation system and data coloring for protecting cloud resources from
Resource provisioning, virtualization, management, and user interfaces
Clients
Service catalogs
Security and performance monitoring

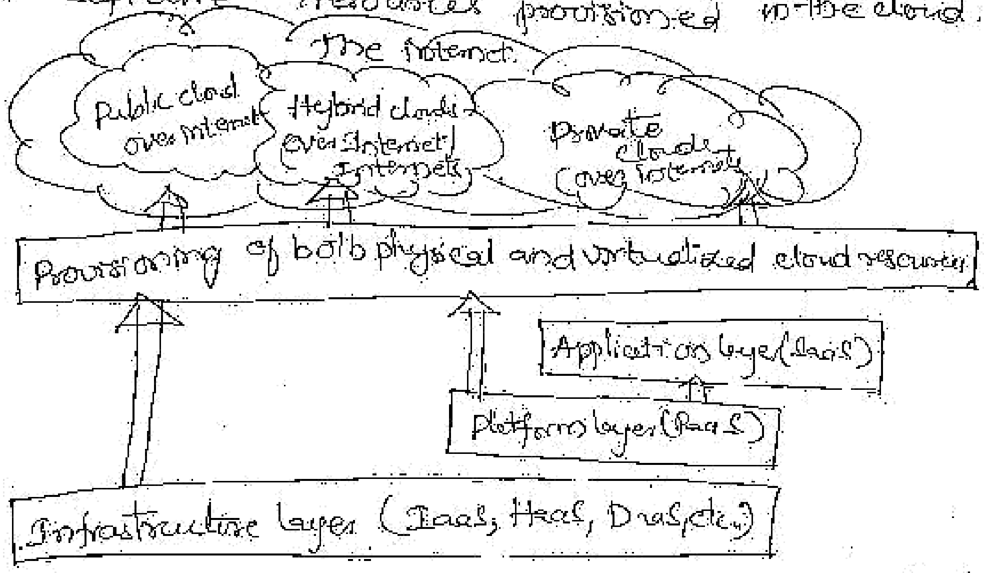
A security aware cloud platform built with a virtual cluster of VMs, storage, and networking resources over the datacenter servers operated by providers.

→ Security becomes a critical issue in safeguarding the operation of all cloud types. The cloud physical platform builder is more concerned about the performance / price ratio and reliability issues than sheer speed performance.

Layered cloud Architectural Development:-

→ The architecture of a cloud is developed at three layers: infrastructure, platform, and application, as

→ These three development layers are implemented with virtualization and standardization of hardware and software resources provisioned in the cloud.



The Internet
Public cloud over Internet
Hybrid clouds over Internet/ Intranets
Private clouds (over Internet)
Provisioning of both physical and virtualized cloud resources
Application layer (SaaS)
Platform layer (PaaS)
Infrastructure layer (IaaS, HaaS, DaaS, etc.)

Layered architectural development of the cloud platform for IaaS, PaaS, and SaaS applications over the Internet.

→ The services to public, private, and hybrid clouds are conveyed to users through networking support over the Internet and intranets involved.

→ It is clear that infrastructure layer deployed first to support IaaS services. It is the foundation for building the platform layer the cloud for supporting PaaS services.

→ The platform layer a foundation for implementing the application layer for SaaS applications.

→ This layer provides users with an environment to develop their applications.

→ In a way, the virtualized cloud platform serves as a "System middleware" between the infrastructure and application layers of the cloud.

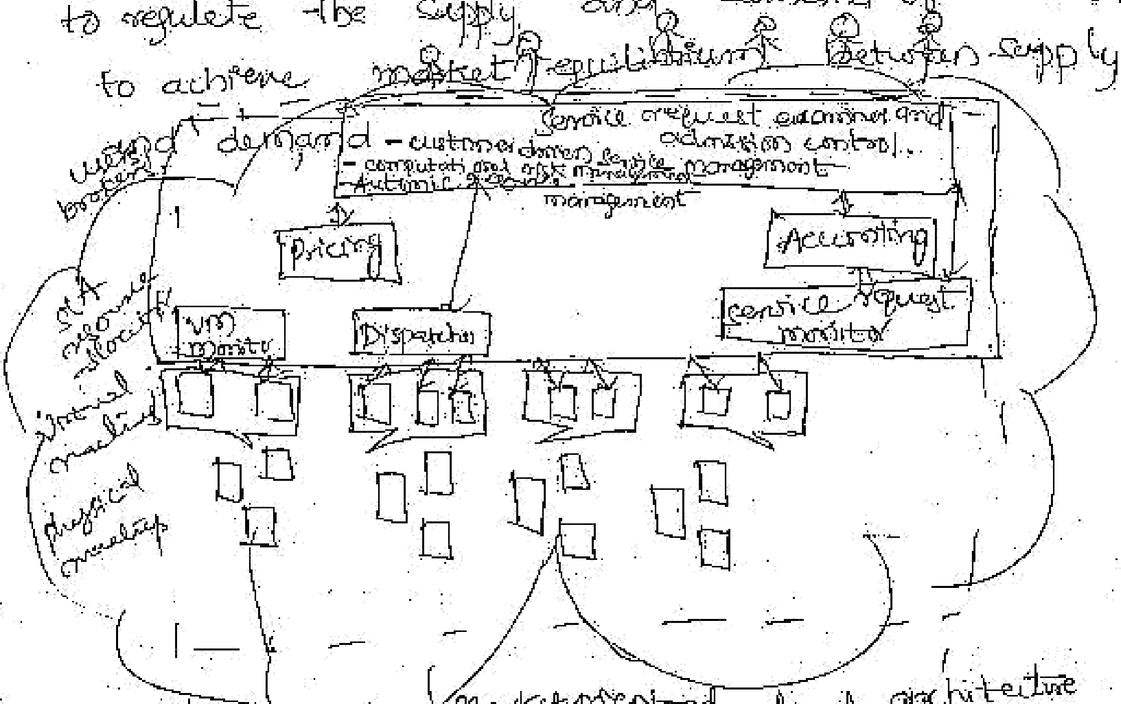
→ The application layer is formed with a collection of all needed software modules for SaaS applications

→ This layer provides the services like CRM, Email, financial transactions, supply chain management.

Market oriented cloud Architecture :—

→ As consumers rely on cloud providers to meet more of their computing needs, they will require a specific level of QOS to be maintained by their providers, in order to meet their objectives and sustain their operations.

→ Market oriented resource management is necessary to regulate the supply and demand of cloud resources to achieve market equilibrium between supply and demand



users brokers
SLA resource allocator
Service request examiner and admission control.
- customer driven service management
- computational risk management
- Autonomic resource management
Pricing
Accounting
VM monitor
Dispatcher
Service request monitor
Virtual machines
physical machines

Market oriented cloud architecture

→ The pricing mechanism decides how service requests are charged. For instance, requests can be charged based on submission time (Peak/off-peak), pricing rates (fixed/ changing), or availability of resources (supply /demand)..

Quality of Service factors :-

→ A data center comprises of multiple computing servers that provide resources to meet service demands.

→ In the case of a cloud a commercial offering to enable crucial business operations of companies, there are critical QoS parameters to consider in a service request, such as time, cost, reliability and trust/ security.

→ QoS requirements cannot be static and may change over time due to continuing changes in business operations and operating environments.

→ The should be greater importance on customers.

→ Since they pay to access services in cloud.

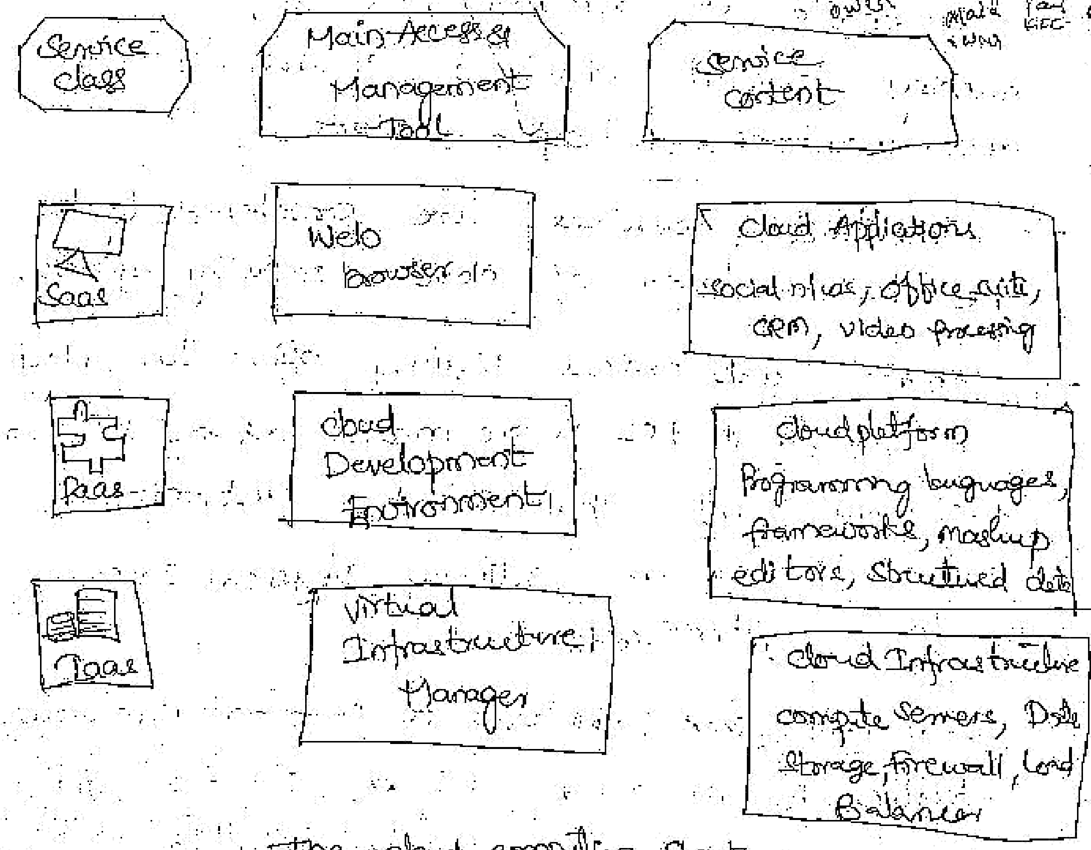
→ So the quality of service is must required in cloud services.

Layers and Types of Clouds

→ cloud computing services are divided into three classes, according to the abstraction level of the capability provided and the service model of providers,

(1) Infrastructure as a service
(2) platform as a service
(3) software as a service

on-premises	I [illegible]	P [illegible]	SaaS
Application			✓
Data		✓	✓
Runtime			✓
Middleware		✓	✓
O/S		✓	✓
Virtualization	✓	✓	✓
Servers	✓	✓	✓
Storage	✓	✓	✓
Networking	✓	✓	✓
	own	Maid [illegible]	Pay KFC



Service class
Main Access & Management Tool
Service content
SaaS
Web browser
Cloud Applications
social n/ws, office suite, CRM, video processing
PaaS
cloud Development Environment
cloud platform
Programming languages, frameworks, mashup editors, Structured data
IaaS
Virtual Infrastructure Manager
cloud Infrastructure
compute servers, Data storage, firewall, Load Balancer

The cloud computing stack.

→ These abstraction levels can be also viewed(?) as layered architecture where services of higher can be composed from services of the underlying layer.

Infrastructure as a Service:-

→ Offering virtualized resources (computation, storage and communication) on demand is known as Infrastructure as a Service (IaaS).

→ It enables on demand provisioning of servers running several choices of operating systems and a customized software stack.

→ Infrastructure services are considered to be the bottom layer of cloud computing systems.

→ Amazon web services mainly offers IaaS, which in case of its EC2 service means offering VMs with a software stack that can be customized similar to how an ordinary physical server would be customized.

→ Users are given privileges to perform numerous activities to the server, such as: starting and stopping it, customizing it by installing software packages, and attaching virtual disks to it.

platform as a service:

→ Another approach is to offer a higher level abstraction to a make cloud easily programmable, known as platform as a service (Paas).

→ A cloud platform offers an environment on which developers create and deploy applications and do not necessarily need to know how many processors or how much memory that application will be using.

→ In addition, multiple programming models and specialized services (eg, data access, authentication and payments) are offered as building blocks to new applications.

→ Google App Engine, an example of platform as a service, offers a scalable environment for developing and hosting web applications, which should be written in a specific programming languages such as python (or) Java.

Software as a service:—

→ Applications reside on the top of the cloud stack.

→ Services provided by this layer can be accessed by end users through web portals.

→ Therefore, consumers are increasingly shifting from locally installed computer programs to on-line software services that offer the same functionality

→ Traditional desktop applications such as word processing and spreadsheet can now be accessed as a service in the web.

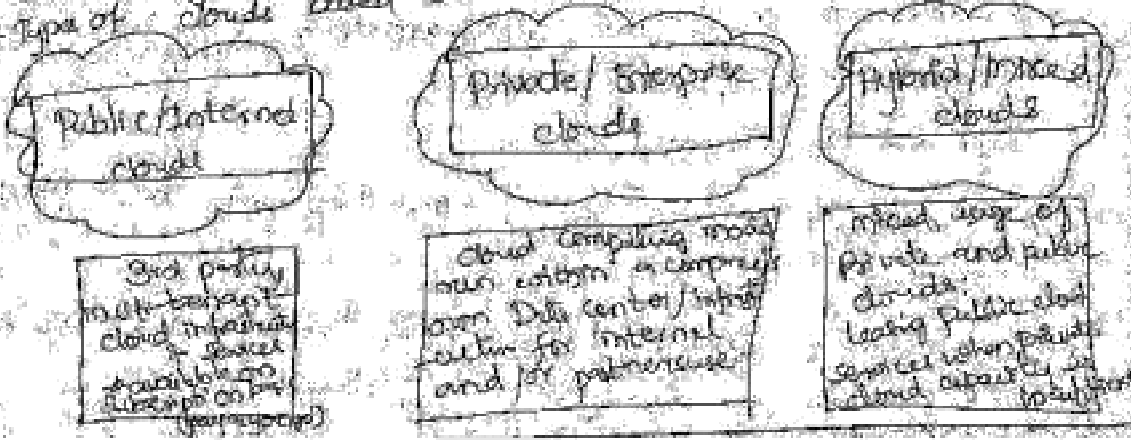
→ This model of delivering applications known as software as a service (SaaS)

→ Sales force.com which relies on the SaaS model, offers business productivity applications (CRM) that reside completely on their servers, allowing customers to customize and access applications on demand

Deployment models:-

→ A cloud can be classified as public, private, community, or hybrid based on model of deployments.

+ Type of clouds based on deployment models



Public/Internet clouds
3rd party multi-tenant cloud infrastructure & services available on (Lease on basis) (pay per usage)
Private/Enterprise clouds
cloud computing model run within a company's own Data center/infrastructure for internal and/or partners use
Hybrid/mixed clouds
mixed usage of private and public clouds: leasing public cloud services when private cloud capacity is insufficient

Grid vs. cloud

→ Grid computing is a network based computational model that has the ability to process large volumes of data with the help of a group of networked computers that coordinate to solve a problem together.

→ Grid computing	Cloud computing
It is for Application Oriented	→ It is for Service Oriented
→ The resources are distributed among different computing units for processing a single task.	→ The computing resources are managed centrally and are placed over multiple servers in clusters
→ Grids are generally owned and managed by an organisation within its premises.	→ The cloud servers are owned by infrastructure providers and are placed in physically disparate locations.
→ It operates within a corporate network.	→ It can also be accessed through the Internet
→ It provides a shared pool of computing resources on an as needed basis.	→ It involves dealing with a common problem using varying number of computer resources
→ It's a collection of interconnected computers and networks that can be called for large-scale processing tasks.	→ more than one computer coordinates to resolve the problem together.

4 → While grid computing involves virtualizing computing resources to store massive amounts of data, whereas cloud computing is where an application doesn't access resources directly, rather it accesses them through a service over the internet.

→ In grid computing, resources are distributed over grids, whereas in cloud computing, resources are managed centrally.

→ The term "cloud" refers to the internet in cloud computing and as a whole it means internet-based computing. The cloud manages data, security, requirements, job queues, etc. by eliminating the needs and complexity of buying hardware and software needed to build applications which are to be delivered as a service over the cloud.

→ Grid computing is mostly used by academic research and is able to handle large sets of limited duration jobs that involve huge volumes of data.

Architectural Design Challenges

→ Challenge1 :- Service Availability and Data Lock-in problem:-

→ The management of a cloud service by a single company is often the source of single points of failure. To achieve (High availability) HA, one can consider using multiple cloud providers. Therefore, using multiple cloud providers may provide more protection from failures.

→ In addition to ~~mitigating~~ reducing data lock-in concerns, standardization of APIs enables a new usage model in which the same software infrastructure can be used in both public and private clouds.

Challenge2 :- Data privacy and security concerns:-

→ current cloud offerings are essentially public (rather than private) networks, exposing the system to more attacks.

→ For example, you could encrypt your data before placing it in a cloud.

→ In a cloud environment, a newer attacks may result from hypervisor malware, guest hopping and hijacking, or VM rootkit.

→ so the security to the data is the main concern

Challenge 3:- Unpredictable performance Bottlenecks:- Point of congestion in a production system (or) occurs when workload is too high [illegible] production phases

→ Multiple VMs can share CPUs and main memory in cloud computing, but I/O sharing is problematic.

→ Internet applications continue to become more data-intensive.
parallel [illegible]

cloud users and providers have to think about the implications of placement and traffic at every level of the system, if they want to minimize costs.

→ This kind of reasoning can be seen in Amazon's development of its new cloud-front service. Therefore, data transfer bottlenecks must be removed, bottleneck links must be widened, and weak servers should be removed.

Challenge 4:- Distributed Storage and widespread Software Bugs:-

→ The database is always growing in cloud applications, the opportunity is to create a storage system that will not only meet this growth, but also combine it with the cloud advantage of scaling arbitrarily up and down on demand.

→ This demands design of efficient distributed storage area networks.

→ Large-scale distributed bugs cannot be reproduced so the debugging must occur at a scale in the production data centers.

Challenge 5:- Cloud scalability, Interoperability, and Standardization:-

Application on one platform should be able to incorporate services from the other platform.

→ The pay-as-you-go model applies to storage and network bandwidth; both are counted in terms of the number of bytes used. GAE automatically scales in response to load increases and decreases; users are charged by the cycles used.

→ AWS charges by the hour for the number of VM instances used, even if the machine is idle.

→ Open virtualization format (OVF) describes an open, source, portable, efficient, and extensible format for the packaging and distribution of VMs.

→ It also defines a format for distributing software to be deployed in VMs.

→ Interms of cloud standardization, the ability for virtual appliances to run on any virtual platform. we also need to enable VMs to run on heterogeneous hardware platform hypervisors.

Challenge 6:- Software Licensing and Reputation Sharing:-

→ Many cloud computing providers originally relied on open source software because the licensing model for commercial software is not ideal for utility computing.

→ One can consider using both pay-for-use and bulk-use licensing schemes to widen the business coverage.

→ One customer's bad behaviour can effect the reputation of the entire cloud.

→ This problem must be solved ~~at the~~ SLA (service level aggrement) level.

Desired features of a cloud

certain features of a cloud are essential to enable services that truly represent the cloud computing model and satisfy expectations of consumers, and cloud offerings must be (i) self service

(ii) per usage-metered and billed.

(iii) elastic

(iv) customizable.

(i) self service

→ consumers of cloud computing services expect on demand, nearly instant access to resources.

→ To support this expectation, clouds must allow self-service access so that customers can request, customize, pay and use services without intervention of human operators.

(ii) Per-usage Metering and Billing

→ Services must be priced on short term basis (eg. by the hour), allowing users to release (and not pay for) resources as soon as they are not needed.

→ for these reasons, clouds must implement features to allow efficient trading of service such as pricing accounting, and billing.

→ Metering should be done accordingly for different types of service (eg. Storage, processing, and bandwidth) and usage promptly reported, thus providing greater transparency.

(iii) Elasticity

→ Cloud computing gives the illusion of infite computing resources available on demand.

→ Therefore users can expect clouds to rapidly provide resources in any quantity at any time.

→

(iv) customization

→ Resources rented from the cloud must be highly customizable.

→ In the case of infrastructure services, customization means allowing users to deploy specialized virtual appliances and to be given privileged (root) access to the virtual servers.

→ Other service classes (PaaS and SaaS) offer less flexibility and are not suitable for general-purpose computing, but still are expected to provide a certain level of customization.

Basic principles of cloud computing :

1. cost/benefit: Evaluate the benefits of cloud based on full understanding of the costs of cloud compared with the costs of other technology platform business solutions.

2. Trust: Make trust an essential part of cloud solutions, building trust into all business process that depend on cloud computing.

3. capability: Integrate the full extent of capabilities that cloud providers offer with internal resources to provide a comprehensive technical support and delivery solution.

4. Enterprise risk: Take an enterprise risk management perspective to manage the adoption and use of cloud.

5. Accountability: Manage accountabilities by clearly defining internal and provider responsibilities.

cloud computing is different from your traditional web service because of the principles behind cloud computing These principles are :

Resource pooling: cloud computing providers harness large economies of scale through resource pooling. They put together a vast network of servers and hard drives and apply the same set of configurations, protection and the works for them.

Virtualization: Users do not have to care about the physical states of their hardware nor worry about hardware compatibility.

Elasticity: Addition of more hard disk space or server bandwidth can be done with a few clicks of the on-demand. Geographical scalability is also available in cloud computing. One can choose to replicate data to several data centers around the world.

Automatic / easy resource deployment:- The user only needs to choose the types and specifications of the resources he require and the cloud computing provider will configure and set them up automatically.

Metered billing: Users are charged for only what they use.

→ These principles allow cloud computing to bring more cost-savings, automation and flexibility to the users, compared to using a traditional webservice provider.

Challenges and Risks

→ Despite the initial success and popularity of the cloud computing paradigm and extensive availability of providers and tools, a significant number of challenges and risks are inherent to this new model of computing.

→ Providers, developers, and end users must consider these challenges and risks to take good advantage of cloud computing.

→ Issues to be faced include user privacy, data security, data-lock-in, availability of service, disaster recovery, performance, scalability, energy-efficiency, and programmability.

1. Security, Privacy and Trust:

→ Armbrust et al. cite information security as a main issue: "current cloud offerings are essentially public... exposing the system to more attacks." For this reason there are potentially additional challenges to make cloud computing environments are secure as in-house IT systems.

→ Security and privacy affect the entire cloud computing stack, since there is a massive use of third-party services and infrastructures that are used to host important data or to perform critical operations

→ In this scenario, the trust toward providers is fundamental to ensure the desired level of privacy for applications hosted in the cloud.

2. Data Lock-In and Standardization

→ A major concern of cloud computing users is about having their data locked-in by a certain provider.

→ users may want to move data and applications out from a provider that does not meet their requireme-nts.

→ However, in their current form, cloud computing infrastru-ctures and platforms do not employ standard methods of storing and user data and applications.

→ Consequently, they do not interoperate and user data are not portable

→ The answer to this concern is standardization. In this direction, there are efforts to create open standards for cloud computing.

3. Availability, Fault-Tolerance, and Disaster Recovery-

→ It is expected that users will have certain expectations about the service level to be provided once their applications are moved to the cloud.

→ The expectations include availability of the service its overall performance, and what measures are to be taken when something goes wrong in the system or its components.

Resource Management and Energy - Efficiency :-

→ Resource Management and Energy - Efficiency :-

→ One important challenge faced by providers of cloud computing services is the efficient management of virtualized resource pools. Physical resources such as CPU cores, disk space, and network bandwidth must be sliced and shared among virtual machines running potentially heterogeneous workloads.

→ Data centers consumes large amounts of electricity.

→ According to a data published by HP, 100 server racks can consume 1.3 MW of power and another 1.3 MW are required by the cooling system -.

→ To optimize application performance, dynamic resource management can also improve utilization and consequently minimize energy consumption in data centers.

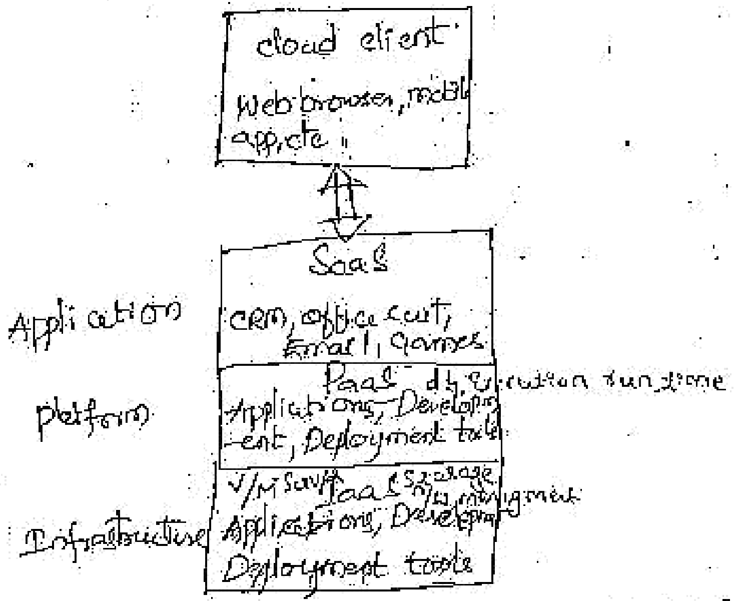
Service models

→ cloud computing based on service model.

categories of service model.

The service models are categorized into three basic models:

1. Software as a service (SaaS)
2. platform as a service (PaaS)
3. Infrastructure-as-a-service (IaaS)



cloud client
Web browser, mobile app, etc.
SaaS
CRM, office suit, Email, Games
PaaS
Applications, Development, Deployment tools
IaaS
V/M server, storage, management
Applications, Development, Deployment tools
Application
platform
Infrastructure
dy. execution run time

* Service models in cloud computing *

1) Software as a Service (SaaS)

→ SaaS is known as 'on-demand software'.

→ It is a software distribution model. In this model, the applications are hosted by a cloud service provider and publicized to the customers over internet.

→ Associated data and software are hosted centrally on the cloud server in SaaS.

→ User can access SaaS by using a web browser.
Ex: CRM, Office suit, Email, games etc. are the software applications.

→ The companies like Google, Microsoft provide their applications as a service to the end users.

Advantages of SaaS:-

→ SaaS is easy to buy because the pricing of SaaS is based on monthly or annual fee and it allows the organizations to access business functionalities at a small cost, which is less than licensed applications.

→ SaaS needed less hardware, because the software is hosted remotely, hence organizations do not need to invest in additional hardware.

→ Less maintenance cost is required for SaaS and not require special software or hardware versions.

Disadvantages of SaaS:-

→ SaaS applications are totally dependent on Internet connection. They are not usable without Internet connection.

→ It is difficult to switch amongst the SaaS vendors.

2> Platform-as-a-Service (PaaS)

→ PaaS is a programming platform for developers. This platform is generated for the programmers to create, test, run and manage the applications.

→ A developer can easily write applications and deploy it directly into PaaS layer.

→ PaaS gives the runtime environment for application development and deployment tools.

→ Google App Engine (GAE), windows Azure, Salesforce.com are the Examples of PaaS.

Advantages of PaaS:-

→ PaaS is easier to develop. Developer can concentrate on the development and innovation without worrying about the infrastructure.

→ In PaaS, developer only requires a PC and an Internet connection to start building applications.

Disadvantages of PaaS:-

→ One developer can write the applications as per the platform provided by PaaS vendor hence the moving the application to another PaaS vendor is a problem.

3) Infrastructure-as-a-Service (IaaS)

→ IaaS is a way to deliver a cloud computing infrastructure like server, storage, network and operating system.

→ The customers can access these resources over cloud computing platform i.e., Internet as an on demand service.

→ In IaaS, you buy complete resources rather than purchasing server, software, data center space or network equipment.

Advantages of IaaS :-

→ In IaaS, user can dynamically choose a CPU, memory storage configuration according to need.

→ Users can easily access the vast computing power available on IaaS cloud

Disadvantages of IaaS :-

→ IaaS cloud computing platform model is dependent on availability of Internet and Virtualization services.

Implementation levels of Virtualization:

→ Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine.

→ The idea of VMs can be dated back to the 1960s.

→ the purpose of a VM is to enhance resource sharing by many users and improve computer performance interms of resource utilization and application flexibility.

→ Hardware resources (CPU, memory, I/O devices etc.) or software resources (operating systems and system libraries) can be virtualized in various functional layers.

Levels of virtualization implementation:

→ After virtualization, different user applications managed by their own operating systems (guest OS) can run on the same hardware, independent of the host OS.

→ This is often done by adding additional software, called a virtualization layer.

→ This virtualization layer is known as hypervisor or virtual machine monitor (VMM).

→ In VMs the applications run with their own guest OS over the virtualized CPU, memory and I/O resources.

→ The main function of the software layer for virtualization is to virtualize the physical hardware of a host machine into virtual resources to be used by the VMs.

→ This can be implemented at various operational levels, the virtualization software creates the abstraction of VMs by interposing a virtualization layer at various levels of a computer system.

The common virtualization layers include

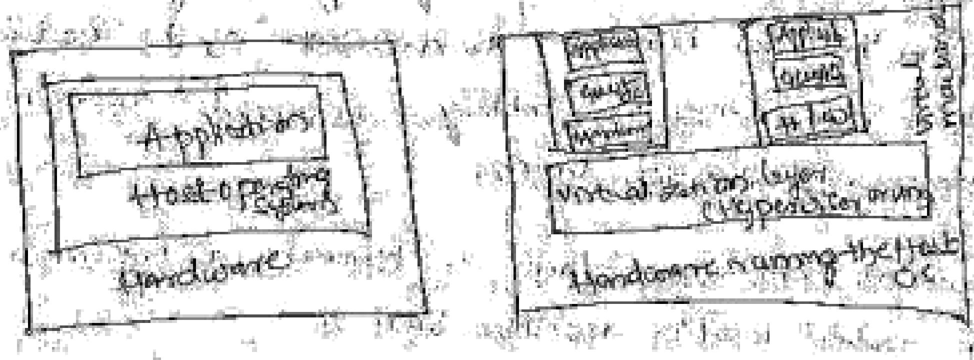
1) Instruction set architecture (ISA) level.

2) Hardware level

3) Operating system level.

4) library support level

5) application level



The architecture of computer system before and after virtualization.

Virtulazation =>

→ On Computer Running Android - blue Stack Emulator

→ Running another O.S or on only one M/C

On Window's Running Ubantu.

→ On 1 M/C running 2-3 O.S Running

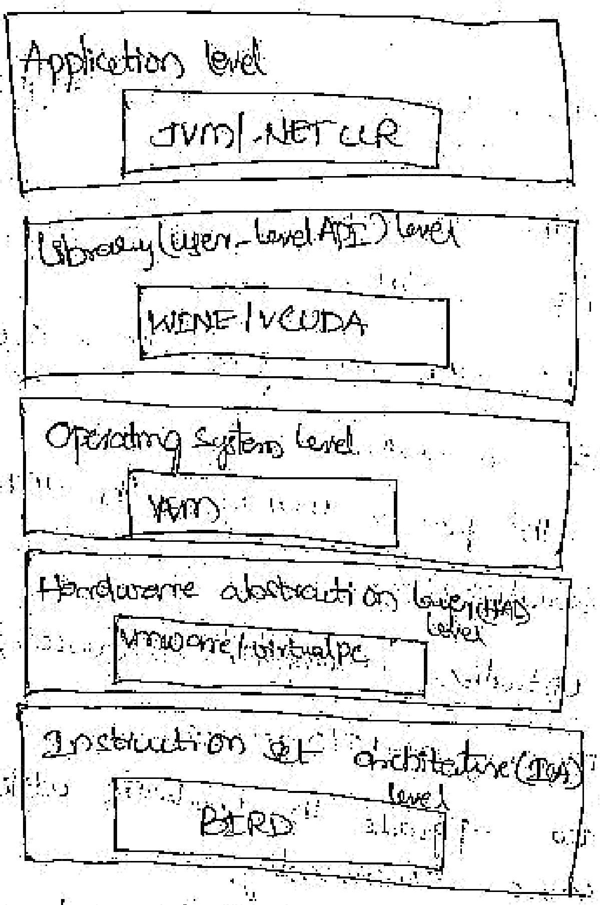
→ Virtualization is abstraction of Computer Resources

→ Creation of virtual Resources
like Server
N/W
file

→ Harddisk doing partitization
not doing - cut.

→ Benefit =>
(I) Co-Existance of O.S on Same M/C
Virtual M/C Ubantu, Kali Linux
On virtual M/C.

(II) Protection
- Code if malliscious. if we ~~Ro~~ are Running in Main M/C then it ll become problem S/m ll get clash so we Ras Run like that code on virtual M/C
If going to clash o.s that virtual os only ntg prob on your Real M/C


Application level
JVM/.NET CLR
Library (user-level API) level
WINE/VCUDA
Operating system level
VEM
Hardware abstraction layer (HAL) level
VMware/virtualpc
Instruction set architecture (ISA) level
BIRD

→ on computer running Android - bluestack ③

- Virtualization ranging from h/w to applications in five abstraction levels.

1) Instruction set Architecture level :-

→ Every machine has an instruction set. This instruction set is an interface between software and hardware. Using this instruction set software communicate with the hardware.

→ so when virtualization carried at this level we can create an emulator the emulator receives all the instructions from the virtual machine then it interprete them and remaps those instructions to the instruction of those hostmachine.

2> <u>Hardware Level</u> :-

→ Hardware level virtualization is performed right on top of the hardware.

→ On the one hand, this approach generates a virtual hardware environment for a VM.

→ On the other hand, this approach generates a virtual hardware environment for a vm.

→ On the other hand, the process manages the underlying hardware through virtualization.

→ The idea is to virtualize a computer's resources, such as its processors, memory, and I/O devices.

→ The intention is to upgrade the hardware utilization rate by multiple users concurrently.

→ The idea was implemented in the IBM VM/370 in the 1960s. More recently, the <u>Xen</u> hypervisor has been applied to virtualize x86-based machines to run Linux or other guest OS applications.

3> <u>Operating System level</u> :-

→ This refers to an abstraction layer between tradition OS and user applications.

→ OS-level virtualization creates <u>isolated containers</u> on a single physical server and the OS instances to utilize the hardware and software in data centers.

→ The containers behave like real servers.

→ OS-level virtualization is commonly used in creating virtual hosting environments to allocate hardware resources among a large number of mutually distrusting users.

→ It is also used, to a lesser extent, in consolidating server hardware by moving services on separate hosts into a containers (or) VMs on one server.

4. Library Support Level.

→ Most applications use APIs exported by user-level libraries rather than using lengthy system calls by the OS.

→ Since most systems provide well-documented APIs, such as an interface becomes another candidate for virtualization.

→ Virtualization with library interface is possible by controlling the communication link between applications and the rest of a system through API hooks.

→ The software tool WINE has implemented this approach to support windows applications on top of UNIX hosts.

to run on Unix-like OS

Compute Unified Device Architecture

→ Another example is the vCUDA which allows applications executing within VMs to leverage GPU hardware acceleration.

graphical processing unit

~~[illegible]~~

(6)

5. <u>User- Application level:-</u>

→ Virtualization at the application level virtualizes an application as a VM.

→ On a traditional os, an application often runs as a process. Therefore, application level virtualization is also known as process-level virtualization.

→ The most popular approach is to deploy high level language (HLL) VMs.

→ In this scenario, the virtualization layer sits as an application program on top of operating and the layer exports an abstraction of a VM that can run programs written and compiled to a particular abstract machine definition.

→ other forms of application-level virtualization are known as application isolation, application sandboxing, or application streaming.

Virtualization Structures Tools and mechanisms:

→ there are three typical classes of VM architecture.

→ Before virtualization, the operating system manages the hardware. After virtualization layer is inserted between the hardware and the operating system.

→ In such a case, the virtualization layer is responsible for converting portions of the real hardware into virtual hardware.

→ Therefore, different operating systems such as Linux and windows can run on the same physical machine, simultaneously.

→ Depending on the position of virtualization layer, there are several classes of VM architectures, namely the hypervisor architecture, para virtualization, and host-based virtualization.

→ The hypervisor is also known as the VMM (virtual machine monitor). They both perform the same virtualization operations.

Hypervisor and Xen Architecture :-

→ The hypervisor supports hardware-level virtualization on bare metal devices like CPU, memory, disk and network interfaces.

Type1 H/w → H.os → Hypv – v1, v2, v3

→ The hypervisor software sits directly between the physical hardware and its Os.

→ This virtualization layer reffered to as either the VMM or the hypervisor.

→ Depending on the functionality, a hypervisor can assume a <u>micro-kernel</u> architecture (or) monolithic hypervisor architecture.

minimum amount of s/w that can be provided

→ A micro kernel hypervisor includes only the basic and unchanging functions (such as physical memory management and processor scheduling). The device drivers and other changeable components are outside the hypervisor.

→ A monolithic hypervisor implements all the aforementioned functions, including those of the device drivers.

<u>The Xen Architecture:-</u>

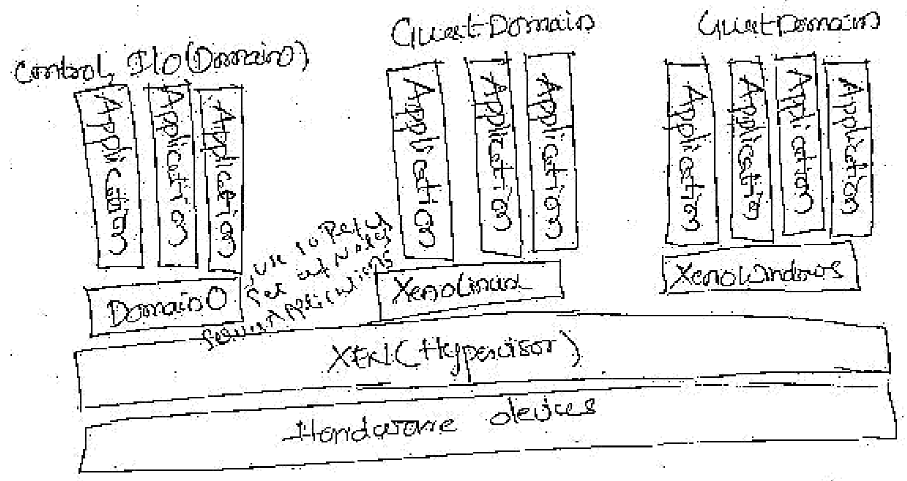
→ Xen is an open source hypervisor program developed by Cambridge University. Xen is a micro kernel hypervisor.

→ Xen doesnot include any device drivers natively.

→ It just provides mechanism by which a guest OS can have direct access to the physical devices.

→ Xen provides a virtual environment located between the hardware and OS.

→ The number of vendors are in the process of developing commercial Xen hypervisors, among them are Citrix Xen Server and Oracle VM.



The Xen architecture's special domain 0 for control and I/O, and several guest domains for user applications.

→ The guest OS, which has control ability, is called Domain 0, and the others are called domain U.

→ Domain 0 is a privileged guest OS of Xen. It is first loaded when Xen boots without any file system drivers being available. Domain 0 is designed to access hardware directly and manage devices.

→ Therefore, one of the responsibilities of Domain0 is to allocate and map hardware resources for the guest domains (the domain U domains).

→ For example, Xen is based on Linux and its management VM is named Domain 0, which has the privilge to manage other VMs implemented on the same host.

[illegible]

→ If Domain 0 is compromised, the hacker can control the entire System.

→ So, in the VM System, security policies are needed to improve the security of Domain 0.

→ Domain 0 is behaving as a VMM, allows users to Create, copy, Save, read, modify, share, migrate and rollback VMs as easily as manipulating a file, which flexibly provides tremendous benefits for users.

OS level virtualizai

i) The OS level virtualization approach doesn't use a hypervisor at all. Instead, the virtualization capability is part of the host OS, which performs all the functions of a fully virtualized hypervisor.

=> The biggest limitation of this approach is that all the guest servers must run the same OS. Each virtual server remains independent from all the others but you can't mix & match operating s/m among them.

Becz all the guest operating s/m must be the same, this is called a homogeneous environment.

Binary Translation with Full virtualization:-

→ Hardware virtualization can be classified into two categories: ① Full virtualization
② Host-based virtualization

→ Full virtualization doesnot need to modify the host OS. It relies on binary translation to trap and to virtualize the execution of certain sensitive, non virtualizable instructions.

→ The guest OSes and their applications consists of noncritical and critical instructions.

→ In a host based system, both a host OS and a guest OS are used.

→ A virtualization software layer is built between the host OS and guest OS.

Full virtualization

→ with full virtualization, non critical instructions run on the h/w directly. While critical instructions are discovered and replaced with traps into the VMM to be emulated by software.

→ Both the hypervisor and VMM approaches are considered full virtualization.

→ Binary translation can incur a large performance overhead. therefore, running noncritical instructions on hardware not only can promote efficiency, but also can ensure system security.

Binary translation of Guest Os Requests using a VMM:-

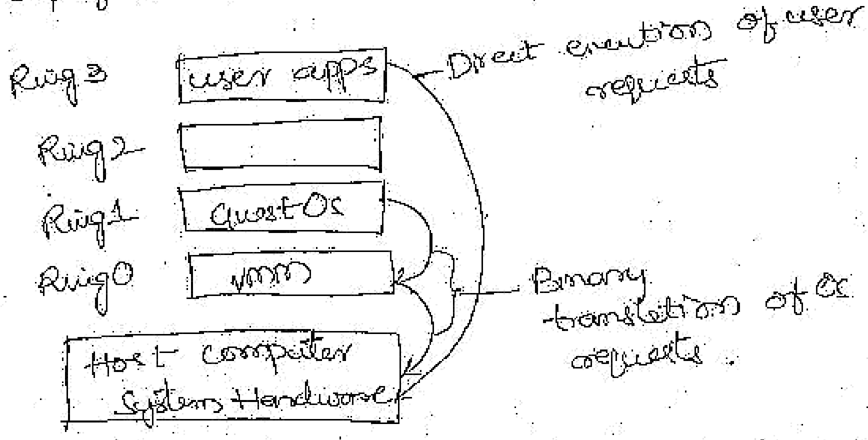
→ This approach was implemented by vmware and many other software companies. Vmware puts the vmm at Ring 0 and guest Os at Ring 1.

→ The vmm scans the instruction stream identifies the privileged, control- and behavior sensitive instruction when these instructions are identified, they are trapped into the vmm. The method used in this emulation which emulates the behavior of these instructions is called "binary translation".

→ Therefore, full virtualization combines binary translation and direct execution, consequently, the guest Os is unaware that is being virtualized. The performance of full virtualization is not ideal because it involve binary translation which is rather time consuming.

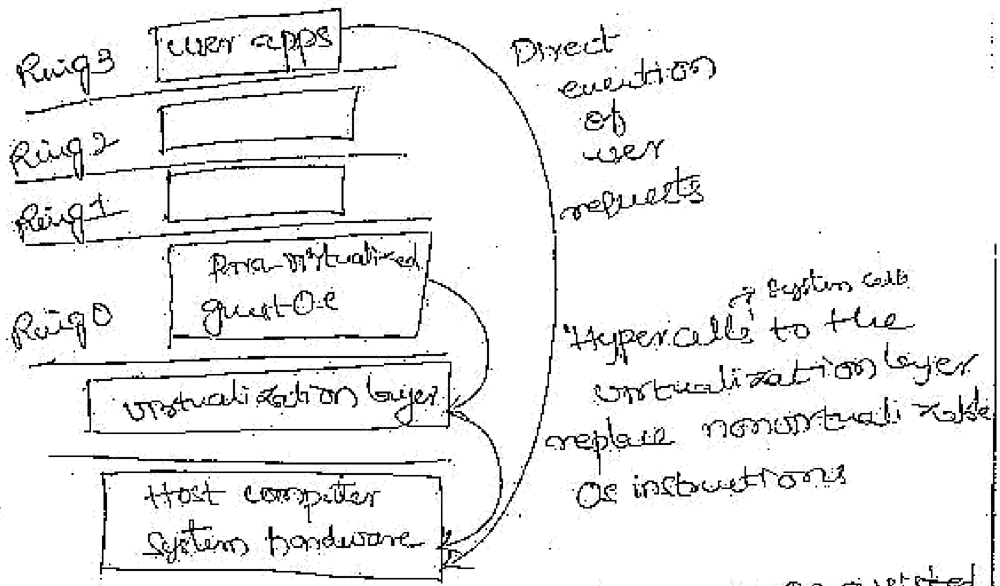
Host based virtualization:-

→ An alternative vm architecture is to install a virtualization layer on top of the host Os.


Ring 3
user apps
Direct execution of user requests
Ring 2
Ring 1
Guest Os
Ring 0
vmm
Binary translation of Os requests
Host computer System Hardware

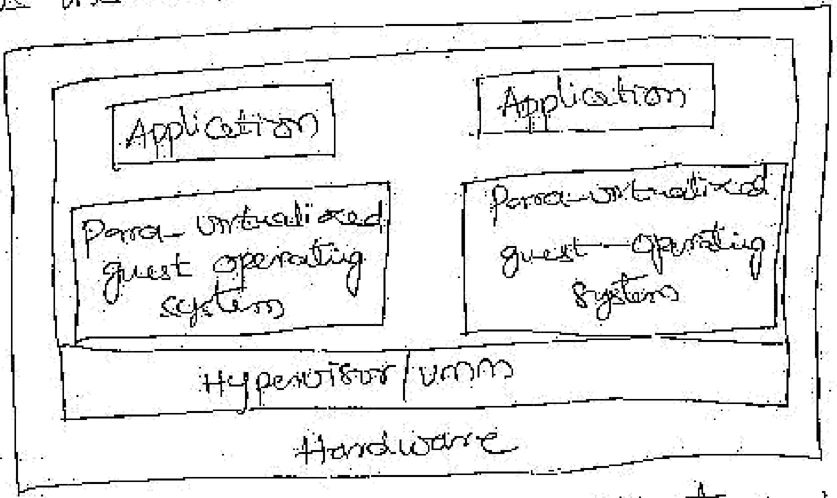
→ This host Os is still responsible for managing the hardware
→ The guest Oses are installed and run on top of the virtualization layer.

Para - virtualization Architecture:-



Ring3
user apps
Ring2
Ring1
Ring0
Para-virtualized guest O.S
virtualization layer
Host computer System hardware
Direct execution of user requests
'Hypercalls' & system calls to the virtualization layer replace nonvirtualizable OS instructions

The use of a para virtualized guest OS assisted by an intelligent compiler to replace nonvirtualizabl OS instructions by hypercalls.



Application
Application
Para-virtualized guest operating system
Para-virtualized guest operating system
Hypervisor / VMM
Hardware

Para-virtualized vm architecture, which involve modifying the guest OS kernel to replace non virtualizable instructions with hypercalls f the hypervisor or the VMM to carryout the virtualization process.

→ Dedicated applications may run on the VMs. Certainly, some other applications can also run with the host OS directly.

→ This host based architecture has some distinct advantages, as enumerated next.

→ First, the user can install this VM architecture without modifying host OS.

→ The virtualizing software can rely on the host OS to provide device drivers and other low level services

→ This will simplify the VM design and ease its deployment

→ Second, the host-based approach appeals to many host machine configurations.

→ the performance of host-based architecture may also be low.

<u>Para-Virtualization with compiler Support</u>:-

Para-virtualization needs to modify the guest operating system. A para-virtualized VM provides special APIs requiring substantial OS modifications in user applications. However, para-virtualization attempts to reduce the virtualization overhead, and thus improve performance by modifying only the guest OS kernel.

→ The traditional X86 processor offers four instruction execution rings: Ring 0, 1, 2, and 3. The lower the ring number, the higher the privilege of instruction being executed.

→ The OS is responsible managing the hardware and the privileged instructions to execute at Ring 0, while user-level applications run at Ring 3.

→ When X86 processor is virtualized, a virtualization layer is inserted between the hardware and the OS. The virtualization layer should also be installed at Ring 0.

→ Para-virtualization replaces non virtualizeable instructions with hypercalls that communicate directly with the hypervisor or VMM. However, when the guest OS kernels modified for virtualization, it can no longer run on the hardware directly.

→ Compared to full virtualization, para virtualization is relatively easy and more practical. The main problem in full virtualization, para is its low performance in binary translation. Therefore, many virtualization products employ the para-virtualization architecture. The popular Xen, KVM (Kernel-Based VM), and VMware ESX are good examples.

KVM (Kernel-Based VM):-

→ This is a linux para-virtualization system - a part of the linux version 2.6.20 kernel.

→ Memory management and scheduling activities are carried out by the existing linux kernel. The KVM does the rest, which makes it simpler than the hypervisor that controls the entire machine.

→ KVM is a hardware-assisted para-virtualization tool, which improves performance and supports unmodified guest OSes such as windows, linux, Solaris and other Unix variants.

Para-virtualization with compiler support:-

→ Unlike the full virtualization architecture which intercepts and emulates privileged and sensitive instructions at runtime, para-virtualization handle these instructions at compile time.

→ The guest OS kernel is modified to replace the privileged and sensitive instructions with hypercal to the hypervisor or VMM. Xen assumes such a paravirtualization architecture.

→ The privileged instructions are implemented by hypercal to the hypervisor. After replacing the instructions with hypercalls, the modified guest OS emulates the behavior of the original guest OS.
hypercalls apply a dedicated service routine in Xen.

Virtualization of CPU, Memory, AND I/O Devices:

→ To support virtualization, processors such as x86 employ a special running mode and instructions, known as hardware-assisted virtualization.

→ In this way, the vmm and guest OS run in different modes and all sensitive instructions of the guest OS and its applications are trapped in the vmm.

Hardware support for virtualization

→ Modern operating systems and processors permit multiple processes to run simultaneously. If there is no protection mechanism in a processor, all instructions from different processes will access the hardware directly and cause a system crash.

→ Therefore, all processors have atleast two modes, user mode and supervisor mode, to ensure controlled access of critical hardware.

→ Instructions running in supervisor mode are called privileged instructions. Other instructions are unprivileged instructions.

→ In a virtualized environment, it is more difficult to make OSes and applications run correctly because there are more layers in the machine stack.

CPU virtualization:-

→ A VM is a duplicate of an existing computer system in which a majority of the VM instructions are executed on the host processor in a native mode.

→ Thus, unprivileged instructions of VMs run directly on the host machine for higher efficiency.

→ Other critical instructions should be handled carefully for correctness and and stability.

→ The critical instructions are divided into three categories: 1> privileged instructions,

2> control sensitive instructions

3> Behaviour-sensitive instructions.

→ Privileged instructions execute in a privileged mode and will be trapped if executed outside this mode.

→ control-sensitive instructions attempt to change the configuration of resources used.

→ Behaviour-sensitive instructions have different behaviours depending on the configuration of resources, including load and store operations over the virtual memory.

→ A CPU Architecture is virtualizable if it supports the ability to run the VM's privileged and unprivileged instructions in the CPU's user mode while VMM runs in the supervisor mode.

→ When the privileged instructions including control-and behavior-sensitive instructions of a VM are executed, they are trapped in the VMM. In this case, the VMM acts as a unified mediator of hardware access from different VMs to guarantee the correctness and stability of the whole systems.

→ However, not all CPU architectures are virtualizable. RISC CPU architectures can be naturally virtualized because all control-and behavior-sensitive instructions are privileged instructions. On the contrary, x86 CPU architectures are not primarily designed to support virtualization.

Hardware-Assisted CPU virtualization:- This technique attempts to simplify virtualization because full or paravirtualization is complicated. Intel and AMD add an additional mode called privilege mode level (some people call it Ring-1) to x86 processors. Therefore, operating systems can still run at Ring 0 and the hypervisor can run at Ring-1.

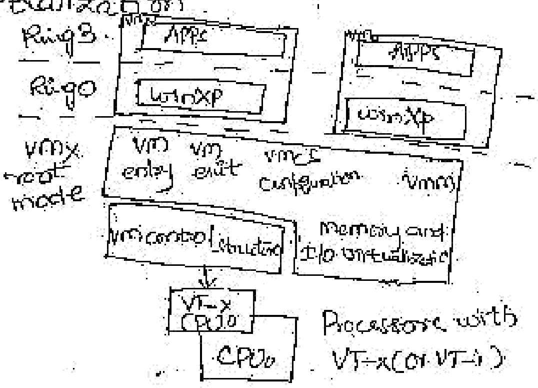
→ All the privileged and sensitive instructions are trapped in the hypervisor automatically. This technique removes the difficulty of implementing binary translation of full virtualization. It also lets the operating system run in VMs without modification.

Ex: Intel Hardware-Assisted CPU virtualization:-

→ Although x86 processors are not virtualizable primarily great effort is taken to virtualize them. They are used widely in comparing RISC processors that the bulk of x86-

Processors is detailed in the following sections.

Intel VT-x technology is an example of hardware-assisted virtualization



CPU state for VMs, a set of additional instructional is added. At the time of this wring, Xen, VMware, and the Microsoft virtual PC all implement their hypervisore by using the VTx technology.

Intel hardware-assisted CPU virtualization.

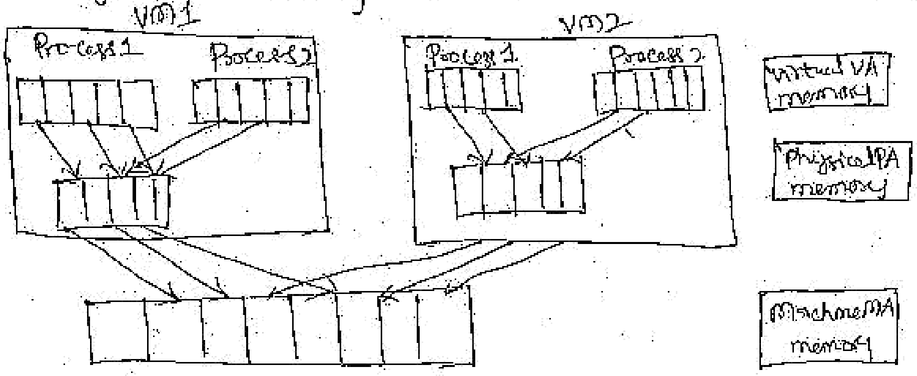
→ Intel calls the privilege level of x86 processors the VMX Root Mode. In order to control the ~~start~~ start and stop of a VM and allocate a memory page to maintain.

→ Generally, hardware-assisted virtualization should have high efficiency. However, since the transition from the hypervisor to the guest OS incurs high overhead switches between processor modes. It sometimes cannot outperform binary translation.

→ Hence, virtualization system such as VMware now use a hybrid approach, in which a few tasks are offloaded to the hardware but the rest is still done in software.

→ In addition, para virtualization and hardware-assisted virtualization can be combined to improve the performance further.

→ The VMM is responsible for mapping the guest physical memory to the actual machine memory.



Two-level mapping procedure.

→ Since each page table of the guest OS has a separate page table in the VMM corresponding it, the VMM page table is called the shadow page table.

→ The MMU handles virtual-to-physical translations as defined by OS.

→ Then the physical memory addresses are translated to machine addresses using another set of page tables defined by the hypervisor.

→ Since modern operating systems maintain a set of page tables for every process, the shadow page tables will get flooded. consequently, the performance overhead and cost of memory will be very high.

→ VMware uses shadow page tables to perform virtual memory-to-machine-memory address translation.

Memory Virtualization:-

→ Virtual memory virtualization is similar to the virtual memory support provided by modern operating systems.

→ In a traditional execution environment, the operating system maintains mapping of virtual memory to machine memory using page tables, which is a one-stage mapping from virtual memory to machine memory.

→ All modern x86 CPUs include a memory management unit (MMU) and a translation lookaside buffer (TLB) to optimize virtual memory performance.

→ Virtual memory virtualization involves sharing the physical system memory in RAM and dynamically allocating it to the physical memory of the VMs.

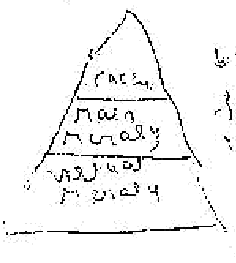
→ That means a two-stage mapping process should be maintained by the guest OS and the VMM, respectively: virtual memory to physical memory and physical memory to machine memory.

→ The guest OS continues to control the mapping of virtual addresses to the physical memory addresses of VMs.

→ But the guest OS cannot directly access the actual machine memory.

→ Processors use TLB hardware to map the virtual memory directly to the machine memory to avoid the two levels of translation on every access.

→ when the guest OS changes the virtual memory to a physical memory mapping, the VMM updates the shadow page tables to enable a direct lookup.



L_1 → memory cache that is directly built into the microprocessor, which is used for storing the microprocessor's recently accessed information, thus it is also called primary cache,

→ also referred to as the internal cache or s/m cache.

→ Cache miss → is an event in which a s/m or application makes a request to retrieve data from a cache, but that specific data is not currently in cache memory.

A cache miss requires the system or application to make a second attempt to locate the data, this time against the slower main database

Cache = is H/w or S/w component that stores data so that future requests for that data can be served faster, the data stored in a cache might be the result of an earlier computation or a copy of data stored elsewhere so that future requests for that data can be served faster.

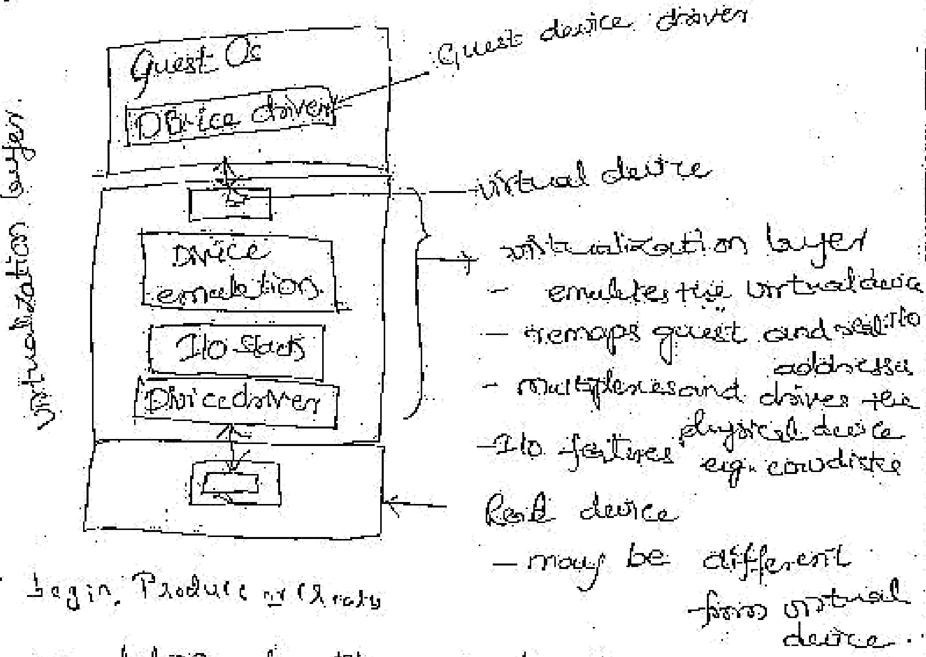
A cache is a reserved storage location that collects temporary data to help websites, browsers, & apps load faster. whether it's a computer, laptop or phone, web browser or app, you'll find some variety of a cache. A cache makes it easy to quickly retri[illegible]

I/O Virtualization :-

→ I/O virtualization involves managing the routing of I/O requests between virtual devices and the shared physical hardware.

→ there are three ways to implement I/O virtualization :
1> Full device emulation
2> Para-virtualization
3> direct I/O.

→ full device emulation is the first approach for I/O virtualization.



Guest Os
Device driver
Guest device driver
virtual device
Device emulation
I/O stack
Device driver
virtualization layer
→ virtualization layer
- emulates the virtual device
- remaps guest and real I/O address
- multiplexes and drives the physical device
- I/O features eg. cow disks
Real device
- may be different from virtual device.

spawned - begin, Produce or Create

Device emulation for I/O virtualization implemented inside the middle layer that maps real I/O devices into the virtual devices for the guest device to use.

Full device approach the all functions of a device such as device emuration, identification, interrupts, and DMA, are replicated in software.

→ This software is located in the VMM and acts as a virtual device. The I/O acess requests of the guest OS are trapped in the VMM which interacts with the I/O devices.

→ The para-virtualization method of I/O virtualization is typically used in Xen. It is also known as split driver model consisting of a frontend driver and backend driver.

→ The frontend driver is running in Domain U and the backend driver is running in Domain 0.

→ The frontend driver manages the I/O requests for the guest OSes and backend driver is responsible for managing the real I/O devices and para-I/O-virtualization achieves better device performance than full device emulation, it comes with a higher CPU overhead.

→ Direct I/O virtualization lets the VM access devices directly. It can achieve close-to-native performance without high CPU costs. Current direct I/O virtualization implementations focus on networking for mainframes.

Virtual clusters and Resource Management :-

→ A physical cluster is a collection of servers (physical machines) interconnected by a physical network such as a LAN.

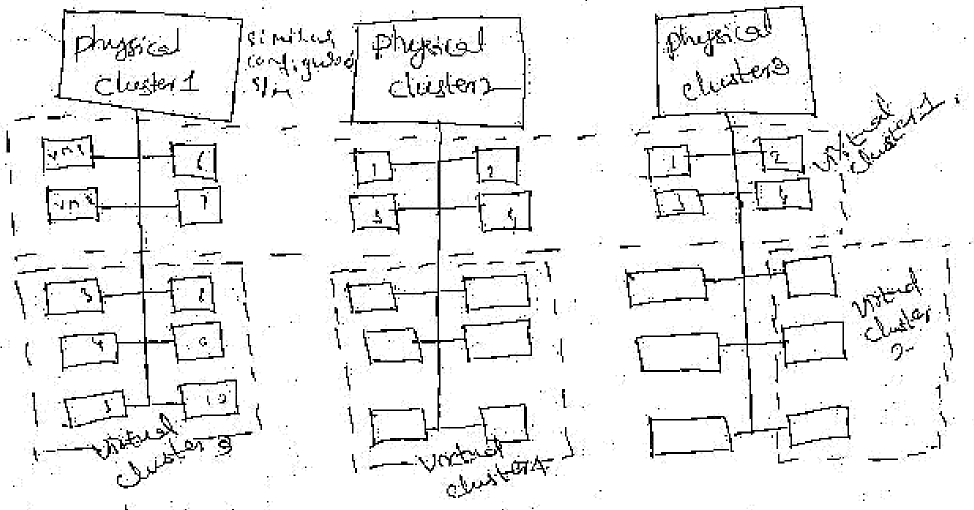
→ Virtual clusters are built with vms installed at distributed servers from one or more physical clusters.

→ The vm in a virtual cluster are interconnected logically by a virtual network across several physical networks.

→ There are three critical design issues of virtual clusters : 1) live migration of vms, 2) memory and file migrations, and dynamic deployment of virtual clusters.

Physical versus virtual clusters :-

→ Each virtual cluster is formed with physical machines where the vm is implemented on a vm hosted by multiple physical clusters.



A cloud platform with four virtual clusters over three physical clusters shaded differently.

→ The properties of virtual clusters :-

→ The virtual cluster nodes can be either physical or virtual machines. Multiple VMs running with different OSes can be deployed on the same physical node.

→ A VM runs with a guest OS, which is often different from the host OS, that manages the resources in the physical machine, where the VM is implemented.

→ The size (number of nodes) of a virtual cluster can grow or shrink dynamically, similar to the way an overlay n/w varies in size in a peer-to-peer (P2P) n/w.

→ The failure of any physical nodes may disable some VMs installed on the failing nodes. But the failure of VMs will not pull down the host system.

Fast Deployment and Effective Scheduling :-

→ The system should have the capability of fast deployment.

→ Here deployment means two things ② to construct ~~image~~ VM and ① distribute software stacks (OS, libraries, applications) to a physical node inside clusters as fast as possible, and to quickly switch runtime environments from one user's virtual cluster to another user's virtual cluster.
live migration of VMs
this applied green computing

→ Mapping VMs onto the most appropriate physical node should promote performance.

→ live migration of VMs allows workloads of one node to transfer to another node
Load balancing using the load index & frequency of user logins can be done to overcome the above issue

→ Dynamically adjusting loads among nodes by live migration of VMs is desired, when the loads on cluster nodes become quite balanced.

(Line is copy of guiding VM)

High - performance Virtual Storage -

It is important to efficiently manage the disk spaces occupied by template software packages. Some storage architectures design can be applied to reduce duplicated blocks in a distributed file system of virtual clusters.

→ Users have their own profiles which store the identification of the data blocks for corresponding VMs in a user - specific virtual clusters.

→ New data blocks are created when users modify the corresponding data.

7. Live VM Migration Steps and Performance Effects :-

→ In a cluster built with mixed nodes of host and guest operating systems, the normal method of operation is to run everything on the physical machine.

→ When a VM fails, its role could be replaced by another VM on a different node, as long as they both run with the same guest OS.

→ In, other words, a physical node can fail over to a VM on another host.

→ This is different from physical-to-physical failover in a traditional physical cluster. The advantage is enhanced failover flexibility.

→ The potential drawback is that a VM must stop playing its role of its residing host node fails.

→ However this problem can be mitigated with VM life migration.

Live migration of a VM consists of the following the steps:-

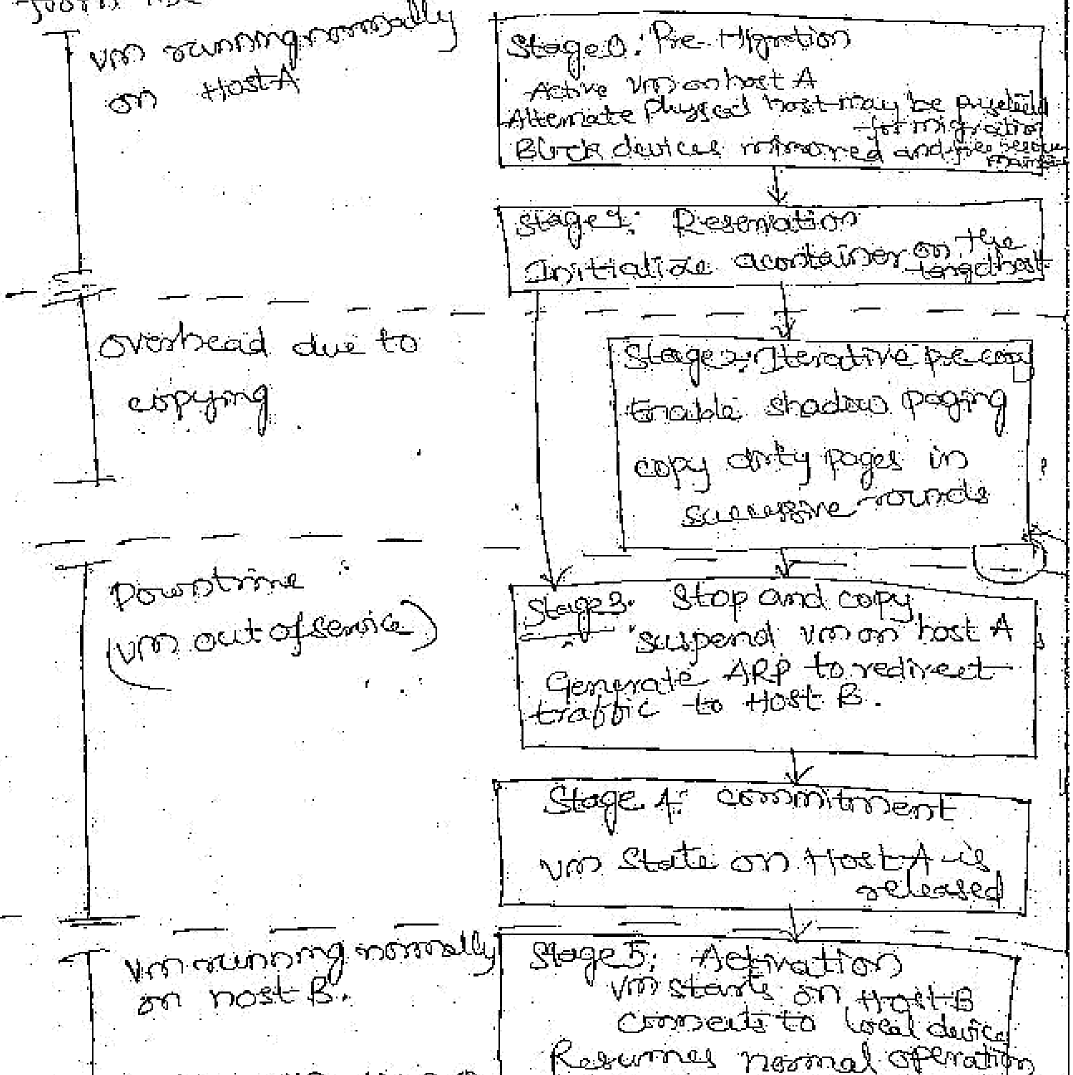
Step 0 and 1; Start migration:- This step makes preparations for the migration, including the migrating VM and destination host.

Step 2; Transfer memory:- Since the whole execution state of VM is stored in memory, Sending the VM's memory to the destination node ensures continuity of the Service provided by VM.

Step 3:- Suspend the VM and copy the last portion of the data:- The migrating VM's execution is Suspended when the last round's memory data is transffered. During this step, the VM is stopped and its applications will no longer run. This "service unavailable" time is called "downtime" of migration.

Step 4 and 5 :- commit and activate the new host :-

After all the needed data is copied, on the destination host, the vm reloads the states and resumes the execution of programs in it, and the service provided by this vm continues. The whole migration process finishes by removing the original vm from the source host.


vm running normally on Host A
Stage 0: Pre-Migration
Active vm on host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
overhead due to copying
Stage 2: Iterative pre-copy
Enable shadow paging
copy dirty pages in successive rounds
Downtime (vm out of service)
Stage 3: Stop and copy
suspend vm on host A
Generate ARP to redirect traffic to Host B.
Stage 4: commitment
vm State on Host A is released
vm running normally on host B.
Stage 5: Activation
vm starts on Host B
Connects to local devices
Resumes normal operation

Live Migration Process of a VM from one host to another

Migration of Memory, files, and Network Resources:-

→ when one system migrates to another physical node, we should consider the following issue.

1 Memory Migration:-

→ This is the one of the most important aspects of VM migration. Memory migration can be in range of hundreds of megabytes to a few gigabytes in a typical system today, and it needs to be done in an efficient manner.

→ The Internet Suspend - Resume (ISR) technique exploits temporal locality this refers to have considerable overlap in the suspended and the resumed instances of a V.M, the memory states differ by the amount of work done since a VM was last suspended before being initiated for migration. Here each file in the system is represented a tree of small subfiles.

→ A copy of this tree exists in both suspended and and resumed VM instances. The advantage of using a tree-based representation of files is that caching ensures the transmission of only those files which have been changed.

2. File System Migration :-

To support vm migration, a system must provide each vm with a consistent, location-independent view of the file system that is available on all hosts.

→ A simple way to achieve this is to provide each vm with its own virtual disk which the file system is mapped to and transport the contents of this virtual disk along with other states of the vm.

→ Due to the current trend of high capacity disks, migration of the contents of entire disk over another is not a viable solution.

→ Another way is to have a global file system across all machines where a vm could be located.

→ This way removes the need to copy files from one machine to another because all files are network accessible.

3. Network Migration :-

→ To enable remote systems to locate and communicate with a vm, each vm must be assigned a virtual IP address known to other entities. This address can be distinct from the IP address of host machine where the vm is currently located.

→ Each vm can also have its own distinct virtual MAC address.

→ The vmm maintains a mapping of the virtual IP and MAC addresses to their corresponding vms. In general, a migrating vm includes all the protocol states and carries its IP address with it.

→ If the source and destination machines of a vm migration are typically connected to a single switched LAN, an unsolicited ARP reply from the migrating host is provided advertising that the IP has moved to a new location.

→ This solves the open network connection problem by reconfiguring all the peers to send future packets to a new location.

Dynamic deployement of virtual clusters:-

The dynamic deployment of four virtual cluster research project has been described and their objectives and reported results.

Project Name	Design Objectives	Reported Results and References
cluster-on-Demand at Duke University	Dynamic Resource allocation with a virtual cluster management system.	sharing of vms by multiple virtual clusters using Sun Grid Engine
Cellular Disco at Stanford University.	To deploy a virtual cluster on a shared-memory multiprocessor	vms deployed on multiple processors under a vmm called Cellular Disco

(3) Project Name	Design objectives	Reported Results and References
VIOLIN at Purdue University	Multiple vm clustering to prove the advantage of dynamic adaption	Reduce execution time of application running VIOLIN with adaption
GRAAL Project at INRIA in France	Performance of parallel algorithms in Xen-enabled virtual clusters	75% max, Performance achieved with 30% resource slacks over vm clusters.

Virtualization for data center Automation

→ Datacenters have grown rapidly in recent years, and all major IT companies are pouring their resources into building new data centers.

→ Datacenter automation means that huge volumes of h/w, s/w, and database resources in these data centers can be allocated dynamically to millions of Internet users simultaneously, with guaranteed QoS and cost-effectiveness.

1. Server consolidation in Datacenters

→ To guarantee that a workload will always be able to cope with all demand levels, therefore it is common that most servers in data centers are underutilized. A large amount of hardware, space, power, and management cost of these servers is wasted.

→ A server consolidation is an approach to improve the low utility ratio of hardware resources by reducing the number of physical servers.

→ Among several server consolidation techniques such as centralized and physical consolidation, virtual-based server consolidation techniques is the most powerful.

→ consolidation enhances hardware utilization. Many underutilized servers are consolidated into fewer servers to enhance resource utilization. Consolidation also facilitates backup services and disaster recovery.

→ This approach enables more agile provisioning and deployment of resources. In a virtual environment, the images of the guest OSes and their applications are readily cloned and reused.

→ The total cost of ownership is reduced. In this sense, server virtualization reduces purchase of new servers, lower maintenance costs, and lower power, cooling and cabling requirements.

→ This approach improves availability and business continuity.

→ To automate data center operations, one must consider resource scheduling, architectural support, power management, automatic (or) autonomic resource management, performance of analytical models, and so on. In virtualized data centers, an efficient, on-demand, fine grained scheduler is one of the key factors to improve resource utilization.

2. Virtual Storage Management :-

Previously,

→ Storage virtualization was largely used to describe the aggregation and repartitioning of disks at very coarse time scales for use by physical machine.

→ In system virtualization, virtual storage includes the storage managed by VMMs and guest OSes.

→ Generally; the data stored in this environment can be classified into two categories;

1) VM images 2) application data.

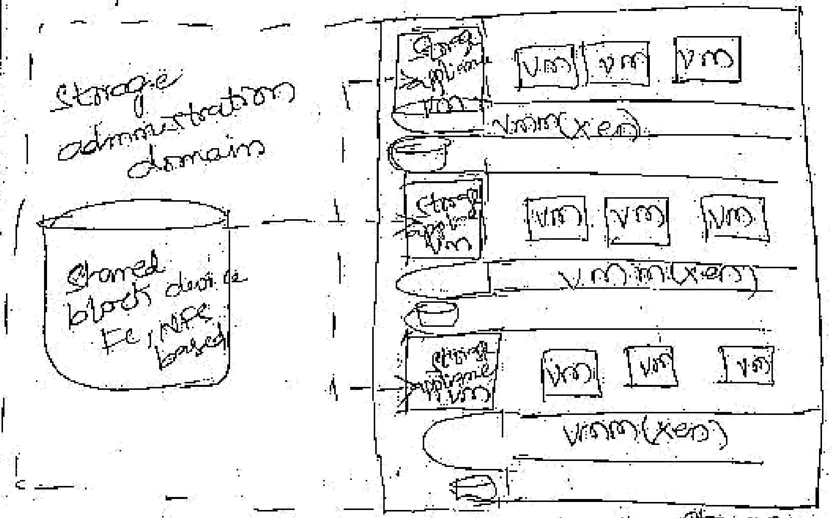
→ VM images are special to the virtual environment while application data includes all other data which is same as the data in traditional OS environments.

→ a virtualization layer is inserted between the hardware and traditional operating systems on a traditional operating system is modified to support virtualization storage operations. it complicates

→ on the otherhand, storage management of the guest OS performs as though it is operating in a real hard disk while the guest OSes cannot access the hard disk directly. on the other hand, many guest OSes content the hard disk while the guest OSes cannot access when many VMs are running on a single physical machine storage management of the underlying VMM is much more complex than that of guest OSes (traditional OSs)

→ In data centers, there are often thousands of vms, which cause the vm images to become flooded. Parallax is a distributed storage system customized for virtualization environments. Content addressable storage (CAS) is a solution to reduce the total size of vm images, and therefore supports a large set of vm-based systems in data centers.

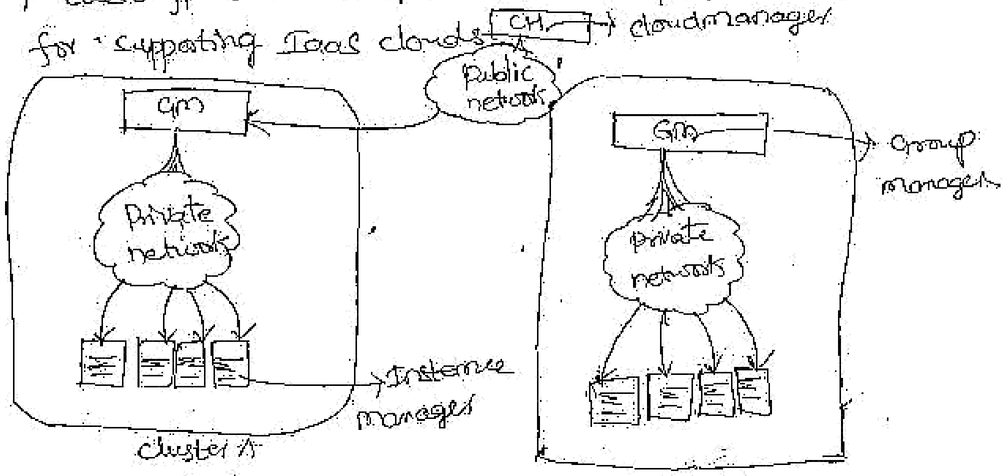
→ Since traditional storage management techniques do not consider the features of storage in virtualization environments, parallax designs a novel architecture in which storage features that have traditionally been implemented



Parallax is a set of per-host storage appliances that share access to a common block device and presents virtual disks to client vms.

→ Nimbus, Eucalyptus and OpenNebula are all open source software available to the general public. Only vSphere 4 is a proprietary OS for cloud resource virtualization and management over data centers.

Ex:- Eucalyptus for virtual Networking of Private cloud.
→ Eucalyptus is an open source s/w program intended mainly for supporting IaaS clouds.



Eucalyptus for building cluster B.
→ Private ~~clouds~~ by establishing networks over the VMs linking through Ethernet and the Internet.

→ Instance Manager controls the execution, inspection, and terminating of VM instances on the host where it runs.

→ GroupManager gathers information about and schedules VM execution on specific instance managers as well as manages virtual instance network.

→ Cloud Manager is the entry-point into the cloud for users and administrators. It queries node managers for information about resources.

3. Cloud Os for Virtualized Data Centers:-

→ Data centers must be virtualized to serve as cloud providers.

Table:- VI managers (Virtual Infrastructure) and Os for virtualizing Data centers:

Manager / Os, platforms License	Resource Being virtualized, web Links	Client API, language	Hypervisors used	Public cloud interface	Special features
Nimbus Linux, Apache V2	VM creation, virtual cluster, www.nimbus project.org/.	EC2 WS, WSRF, CLI	Xen, KVM	EC2	virtual networks
Eucalyptus Linux, BSD	virtual networking www.eucalyptus.com/	EC2 WS, CLI	Xen, KVM	EC2	Virtual networks
OpenNebula Linux, Apache V2	Management of VM, host, virtual network, and scheduling tools, www.opennebula.org/	XML-RPC, CLI, Java	Xen, KVM	EC2, Elastic Host	Virtual networks, dynamic Provision
Vsphere4 Linux, windows, proprietary	virtualizing Os for data centers, www.vmware.com/ products/vsphere/.	CLI, GUI, Portal, WS	VMware ESX, ESXi	VMware Vcloud partners	Data Protection, Vstorage, VMFS, DRM, HA

→ These VI managers and Oses are specially tailored for virtualizing data centers which Often own a large number of servers in clusters.

4. Trust Management in virtualized data Centers :-

→ vmm changes the entire computer architecture.

→ vmm is the base security of virtual system.

→ once a hacker successfully enters the vmm or management vm, the whole system is in danger.

→ A subtler problem arises in protocols that rely on the "freshness" of their random number source for generating session keys.

→ Random number must be "fresh" for security purposes, is reused.

→ With a stream cipher, two different plain texts could be encrypted under the same key stream, which could, in turn, expose both plaintexts could be encrypted if the plaintexts have sufficient redundancy.

VM Based intrusion Detection :-

→ Intrusions are unauthorized access to a certain computer from local or network users and intrusion detection is used to recognize the unauthorized access.

→ Intrusion detection system build on OS. (IDS).

→ A typical IDS can be classified into a host-based IDS (HIDS) or a network based IDS (NIDS) depending on the data source.

→ A-HIDS can be implemented on the monitored system. When the monitored system is attacked by hackers, A-NIDS is based on the flow of networks traffic which can't detect fake actions.

Case Studies: Xen Virtual machine monitors – Xen API

→ The Xen open source virtual machine monitor was originally created through the university of cambridge and developed through Xensource, and citrix systems acquired Xensource in 2007.

→ And Xen technolgy has since emerged in the free edition called XenServer, along with two paid enterprise-class products: Essentials for XenServer Enterprise and Essentials for XenServer platinum.

→ Other commercial implementations for Xen include Oracle VM and Sun xVM.

→ The Xen open source virtual machine monitor is designed for common Intel and IBM architectures, supporting X86 (32 and 64bit), Itanium, and PowerPC based systems.

→ The Xen-hypervisor loads and supports all the subsequent operating systems (Oses) and workloads. This is reffered as a Type 1 (or) bare-metal hypervisor, which runs directly on the system's hardware and hosts Oses above it.

→ There are many operating systems that can serve as a host operating system (domain 0).

→ The hypervisor in the Xen open source virtual machine monitor is particularly noted for its virtual machine live migration capabilities, allowing administrators to move virtual workloads from one physical host to another without shutting down or even disrupting the workload being moved.

→ XenAPI – A Xen project Toolstack that exposes the XenAPI interface. An interface for remotely configuring and controlling virtualized guests running on a Xen enabled host. XenAPI is the core component of Citrix Hypervisor / XenServer. Each toolstack exposes an API, which will run different tools. XenAPI adds additional functionality compared other Xen project toolstacks

→ Extending the software to cover multiple hosts.

→ Facilitating real-time performance monitoring and alerting.

→ Creating upgrade and patching capabilities.

→ Enabling resource pools to include live migration, auto configuration, and disaster recovery.

→ Enhancing the VM lifecycle, VM migration.

vmware :-

→ vmware is a virtualization and cloud computing software provider based in Palo Alto, Calif founded in 1998.

→ vmware its a subsidiary of Dell technologies.

→ Emc corporation originally acquired vmware in 2004 ; Emc was later acquired by Dell Technologies in 2016.

→ vmware bases its virtualization technologies on its bare-metal hypervisor in x86 architecture.

→ A hyper visor is installed on the physical server to allow for multiple virtual machines (vms) to run on the same physical server.

→ Multiple Oses can run on one physical server.

→ All the vms on the same physical server share resources such as networking and RAM.

vmware products :- vmware features :-

vmware products include virtualization, networking and security management tools, software defined datacenter software and storage software.

vmware vsphere is vmware's suite of virtualization products, it is known as vmware infrastructure prior to 2009.

vmware cloud on AWS, customers can run a cluster of vsphere hosts with vSAN and NSX in an Amazon data center and run their workloads.

vmware NSX is a virtual networking and security software offering created when vmware acquired Nicera in 2012.

vmware cloudFoundation is an integrated software stack that bundles vSphere, vmware vSAN and vmware NSX into a single platform.

vmware vSAN is a software based storage feature that is built into the ESXi hypervisor and integrated with vsphere. it pools disk space from multiple ESXi hosts and provisions it via smart polici

vRealize suit allows a user to create and manage hybrid clouds.

vmware Horizon allows organizations to run windows desktops in data centers.

workspace One allows an administrator to control mobile device and cloud hosted virtual desktops.

vmware workstation is the product ever released by s/w company to create and vms directly on a single windows or linux desktop or laptop.

vmware Fusion it is s/w like vmware workstation for mac computers

* Microsoft Virtual Server :-

→ Microsoft Virtual Server is ascalable server virtualization Program distibuted by Microsoft that enables multiple operating systems (OSs) to run on a single physical server.

→ The program operates without third-party device drivers and provides isolation between partitions.

→ Microsoft virtual server is commonly used for server consolidation in business networks and data centers.

→ It can also be used by software developers.

→ Version 2005 R2 can support up to 64 guest OSs and employ symmetric multiprocessing but doesnot virtualize it.

→ In April 2006, microsoft Virtual Server Enterprise Edition became available as a free download.

* Key features of Microsoft Virtual Server :-

→ With Microsoft's virtual server 2005 R2, administrators can manage multiple virtual machines at aglance without having to be physically present at the host server.

Accessibility:-

→ VS2005R2 administration is performed through a web browser rather than through an application interface.

→ Using a web browser interface spares the user from having to install a client application on the user's system.

→ The browser interface is used to create machines, to start and stop them, to change their configuration (memory size, locations of virtual hard disks etc)., and to gain access to the virtual server's console.

→ A user can connect remotely to the server through the browser or through a standalone application called the virtual machine Remote control client (VMRCC). Accessing a remote server through the web interface requires an ActiveX control and the VMRCC is windows only.

Security:-

→ Users or groups can be given varying levels of access to the virtual servers, much as they can have access to files or folders.

→ A given user or group could be allowed to access a server but not to change its properties.

→ connections to the VS2005R2 website should normally be done via SSL. this is especially recommended if you're accessing VS2005R2 from outside a firewall.

Resource allocation :-

→ when you have multiple ~~few~~ virtual servers running on one host, you'll probably want to assign limits on the resources ~~available~~ to each virtual server.

→ When you click on virtual server Resource Allocation and you will see a list of all the available virtual servers, along with the maximum and minimum amount of CPU you can set for each.

→ Each server can also be assigned an index called a "relative weight", from 1 to 10,000 (the default is 100).

→ The higher the weight, the more preference a given server gets interms of resource allocations.

Scripting :-

→ Virtual servers can have command-line scripts assigned to run when certain actions are performed (like turning a machine on or off) or when virtual disk space is low.

→ These scripts can be anything from batch jobs to sophisticated VBscripts that make changes to server configurations via VS2005R2's interface.

CryptDB: Protecting Confidentiality with Encrypted Query Processing

Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan
MIT CSAIL

ABSTRACT

Online applications are vulnerable to theft of sensitive information because adversaries can exploit software bugs to gain access to private data, and because curious or malicious administrators may capture and leak data. CryptDB is a system that provides practical and provable confidentiality in the face of these attacks for applica- tions backed by SQL databases. It works by *executing SQL queries over encrypted data* using a collection of efficient SQL-aware en- cryption schemes. CryptDB can also *chain encryption keys to user passwords*, so that a data item can be decrypted only by using the password of one of the users with access to that data. As a result, a database administrator never gets access to decrypted data, and even if all servers are compromised, an adversary cannot decrypt the data of any user who is not logged in. An analysis of a trace of 126 million SQL queries from a production MySQL server shows that CryptDB can support operations over encrypted data for 99.5% of the 128,840 columns seen in the trace. Our evaluation shows that CryptDB has low overhead, reducing throughput by 14.5% for phpBB, a web forum application, and by 26% for queries from TPC- C, compared to unmodified MySQL. Chaining encryption keys to user passwords requires 11–13 unique schema annotations to secure more than 20 sensitive fields and 2–7 lines of source code changes for three multi-user web applications.

Categories and Subject Descriptors: H.2.7 [**Database Man- agement**]: Database Administration—*Security, integrity, and pro- tection.*

General Terms: Security, design.

1 INTRODUCTION

Theft of private information is a significant problem, particularly for online applications [40]. An adversary can exploit software vulnerabilities to gain unauthorized access to servers [32]; curious or malicious administrators at a hosting or application provider can snoop on private data [6]; and attackers with physical access to servers can access all data on disk and in memory [23].

One approach to reduce the damage caused by server compro- mises is to encrypt sensitive data, as in SUNDR [28], SPORC [16], and Depot [30], and run all computations (application logic) on clients. Unfortunately, several important applications do not lend themselves to this approach, including database-backed web sites that process queries to generate data for the user, and applications that compute over large amounts of data. Even when this approach is tenable, converting an existing server-side application to this form can be difficult. Another approach would be to consider theoret- ical solutions such as fully homomorphic encryption [19], which allows servers to compute arbitrary functions over encrypted data, while only clients see decrypted data. However, fully homomorphic encryption schemes are still prohibitively expensive by orders of magnitude [10, 21].

This paper presents CryptDB, a system that explores an interme- diate design point to provide confidentiality for applications that use database management systems (DBMSes). CryptDB leverages the typical structure of database-backed applications, consisting of a DBMS server and a separate application server, as shown in Figure 1; the latter runs the application code and issues DBMS queries on be- half of one or more users. CryptDB's approach is to *execute queries over encrypted data*, and the key insight that makes it practical is that SQL uses a well-defined set of operators, each of which we are able to support efficiently over encrypted data.

CryptDB addresses two threats. The first threat is a curious database administrator (DBA) who tries to learn private data (e.g., health records, financial statements, personal information) by snoop- ing on the DBMS server; here, CryptDB prevents the DBA from learning private data. The second threat is an adversary that gains complete control of application and DBMS servers. In this case, CryptDB cannot provide any guarantees for users that are logged into the application during an attack, but can still ensure the confi- dentiality of logged-out users' data.

There are two challenges in combating these threats. The first lies in the tension between minimizing the amount of confidential infor- mation revealed to the DBMS server and the ability to efficiently execute a variety of queries. Current approaches for computing over encrypted data are either too slow or do not provide adequate confidentiality, as we discuss in 9. On the other hand, encrypting data with a strong and efficient cryptosystem, such as AES, would prevent the DBMS server from executing many SQL queries, such as queries that ask for the number of employees in the "sales" de- partment or for the names of employees whose salary is greater than
$60,000. In this case, the only practical solution would be to give the DBMS server access to the decryption key, but that would allow an adversary to also gain access to all data.

The second challenge is to minimize the amount of data leaked when an adversary compromises the application server in addition to the DBMS server. Since arbitrary computation on encrypted data is not practical, the application must be able to access decrypted data. The difficulty is thus to ensure that a compromised application can obtain only a limited amount of decrypted data. A na¨ıve solution of assigning each user a different database encryption key for their data does not work for applications with shared data, such as bulletin boards and conference review sites.

CryptDB addresses these challenges using three key ideas:

- The first is to execute SQL queries over encrypted data. CryptDB implements this idea using a *SQL-aware encryption strategy*, which leverages the fact that all SQL queries are made up of a

§

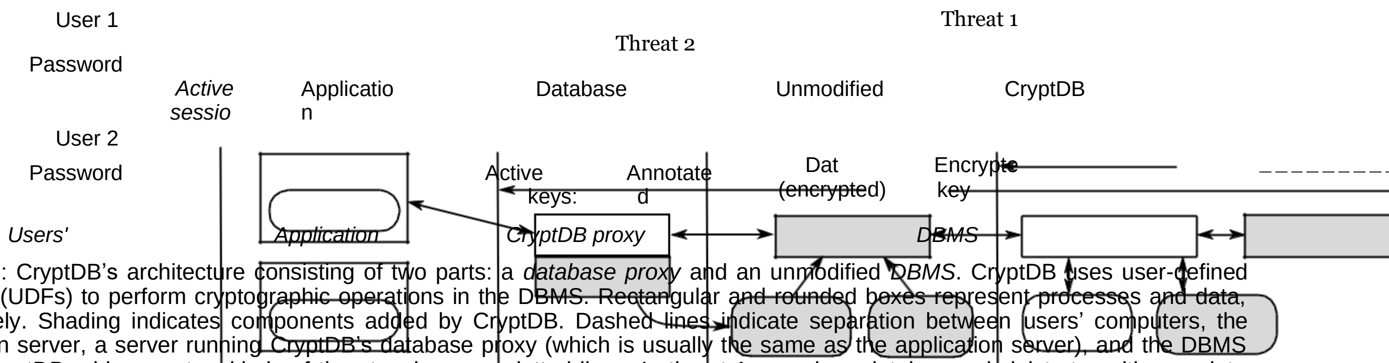


Figure 1 : CryptDB's architecture consisting of two parts: a *database proxy* and an unmodified *DBMS*. CryptDB uses user-defined functions (UDFs) to perform cryptographic operations in the DBMS. Rectangular and rounded boxes represent processes and data, respectively. Shading indicates components added by CryptDB. Dashed lines indicate separation between users' computers, the application server, a server running CryptDB's database proxy (which is usually the same as the application server), and the DBMS server. CryptDB addresses two kinds of threats, shown as dotted lines. In threat 1, a curious database administrator with complete access to the DBMS server snoops on private data, in which case CryptDB prevents the DBA from accessing any private information. In threat 2, an adversary gains complete control over both the software and hardware of the application, proxy, and DBMS servers, in which case CryptDB ensures the adversary cannot obtain data belonging to users that are not logged in (e.g., user 2).

well-defined set of primitive operators, such as equality checks, order comparisons, aggregates (sums), and joins. By adapt- ing known encryption schemes (for equality, additions, and or- der checks) and using a new privacy-preserving cryptographic method for joins, CryptDB encrypts each data item in a way that allows the DBMS to execute on the transformed data. CryptDB is efficient because it mostly uses symmetric-key encryption, avoids fully homomorphic encryption, and runs on unmodified DBMS software (by using user-defined functions).

- The second technique is *adjustable query-based encryption*. Some encryption schemes leak more information than others about the data to the DBMS server, but are required to process certain queries. To avoid revealing all possible encryptions of data to the DBMS *a priori*, CryptDB carefully *adjusts* the SQL-aware encryption scheme for any given data item, depending on the queries observed at run-time. To implement these adjust- ments efficiently, CryptDB uses *onions of encryption*. Onions are a novel way to compactly store multiple ciphertexts within each other in the database and avoid expensive re-encryptions.

- The third idea is to *chain encryption keys to user passwords*, so that each data item in the database can be decrypted only through a chain of keys rooted in the password of one of the users with access to that data. As a result, if the user is not logged into the application, and if the adversary does not know the user's password, the adversary cannot decrypt the user's data, even if the DBMS and the application server are fully compromised. To construct a chain of keys that captures the application's data privacy and sharing policy, CryptDB allows the developer to provide policy annotations over the application's SQL schema, specifying which users (or other principals, such as groups) have access to each data item.

We have implemented CryptDB on both MySQL and Postgres; our design and most of our implementation should be applicable to most standard SQL DBMSes. An analysis of a 10-day trace of 126 million SQL queries from many applications at MIT suggests that CryptDB can support operations over encrypted data for 99.5% of the 128,840 columns seen in the trace. Our evaluation shows that CryptDB has low overhead, reducing throughput by 14.5% for the phpBB web forum application, and by 26% for queries from TPC-C, compared to unmodified MySQL. We evaluated the security of CryptDB on six real applications (including phpBB, the HotCRP conference management software [27], and the OpenEMR medical records application); the results show that CryptDB protects most sensitive fields with highly secure encryption schemes. Chaining encryption keys to user passwords requires 11–13 unique schema annotations to enforce privacy policies on more than 20 sensitive fields (including a new policy in HotCRP for handling papers in conflict with a PC chair) and 2–7 lines of source code changes for three multi-user web applications.

The rest of this paper is structured as follows. In 2, we discuss the threats that CryptDB defends against in more detail. Then, we describe CryptDB's design for encrypted query processing in 3 and for key chaining to user passwords in 4. In 5, we present several case studies of how applications can use CryptDB, and in 6, we discuss limitations of our design, and ways in which it can be extended. Next, we describe our prototype implementation in 7, and evaluate the performance and security of CryptDB, as well as the effort required for application developers to use CryptDB, in §8. We compare CryptDB to related work in §9 and conclude in §10.

2 SECURITY OVERVIEW

Figure 1 shows CryptDB's architecture and threat models. CryptDB works by intercepting all SQL queries in a *database proxy*, which rewrites queries to execute on encrypted data (CryptDB assumes that all queries go through the proxy). The proxy encrypts and decrypts all data, and changes some query operators, while preserving the semantics of the query. The DBMS server never receives decryption keys to the plaintext so it never sees sensitive data, ensuring that a curious DBA cannot gain access to private information (threat 1).

To guard against application, proxy, and DBMS server compro- mises (threat 2), developers annotate their SQL schema to define different *principals*, whose keys will allow decrypting different parts of the database. They also make a small change to their applications to provide encryption keys to the proxy, as described in 4. The proxy determines what parts of the database should be encrypted under what key. The result is that CryptDB guarantees the confi- dentiality of data belonging to users that are not logged in during a compromise (e.g., user 2 in Figure 1), and who do not log in until the compromise is detected and fixed by the administrator.

Although CryptDB protects data confidentiality, it does not ensure the integrity, freshness, or completeness of results returned to the application. An adversary that compromises the application, proxy, or DBMS server, or a malicious DBA, can delete any or all of the data stored in the database. Similarly, attacks on user machines, such as cross-site scripting, are outside of the scope of CryptDB.

We now describe the two threat models

§

addressed by CryptDB, and the security guarantees provided under those threat models.§

§ §

§

2.1 Threat 1: DBMS § Server Compromise

In this threat, CryptDB guards against a curious DBA or other exter- nal attacker with full access to the data stored in the DBMS server. Our goal is confidentiality (data secrecy), not integrity or availability. The attacker is assumed to be *passive*: she wants to learn confidential

§

data, but does not change queries issued by the application, query results, or the data in the DBMS. This threat includes DBMS soft- ware compromises, root access to DBMS machines, and even access to the RAM of physical machines. With the rise in database consol- idation inside enterprise data centers, outsourcing of databases to public cloud computing infrastructures, and the use of third-party DBAs, this threat is increasingly important.

Approach. CryptDB aims to protect data

•

confidentiality against this threat by executing SQL queries over encrypted data on the DBMS server. The proxy uses secret keys to encrypt all data inserted or included in queries issued to the DBMS. Our approach is to allow the DBMS server to perform query processing on encrypted data as it would on an unencrypted database, by enabling it to compute certain functions over the data items based on encrypted data. For example, if the DBMS needs to perform a GROUP BY on column *c*, the DBMS server should be able to determine which items in that column are equal to each other, but not the actual content of each item. Therefore, the proxy needs to enable

•

the DBMS server to determine relationships among data necessary to process a query. By using SQL-aware encryption that adjusts dynamically to the queries presented, CryptDB is careful about what relations it reveals between tuples to the server. For instance, if the DBMS needs to perform only a GROUP BYon a column *c*, the DBMS server should not know the order of the items in column *c*, nor should it know any other information about other columns. If the DBMS is required to perform an ORDER BY, or to find the MAX or MIN, CryptDB reveals the order of items in that column, but not otherwise.

Guarantees. CryptDB provides confidentiality for data content and for names of columns and tables; CryptDB does not hide the overall table structure, the number of rows, the types of columns, or the approximate size of data in bytes. The security of CryptDB is *not perfect:* CryptDB reveals to the DBMS server relationships among data items that correspond to the *classes of computation* that queries perform on the database, such as comparing items for equality, sorting, or performing§ word search. The granularity at which CryptDB allows the DBMS to perform a class of computations is an entire column (or a group of joined columns, for joins), which means that even if a query requires equality checks for a few rows, executing that query on the server would require revealing that class of computation for an entire column. 3.1 describes how these classes of computation map to CryptDB's encryption schemes, and the information they reveal.

More intuitively, CryptDB provides the following properties:

- Sensitive data is never available in plaintext at the DBMS server.

The information revealed to the DBMS server depends on the classes of computation required by the application's queries, subject to constraints specified by the application developer in the schema (§3.5.1):

1. If the application requests no relational predicate filtering on a column, nothing about the data content leaks (other than its size in bytes).
2. If the application requests equality checks on a column, CryptDB's proxy reveals which items repeat in that column (the histogram), but not the actual values.
3. If the application requests order checks on a column, the proxy reveals the order of the elements in the column.

The DBMS server cannot compute the (encrypted) results for queries that involve computation classes not requested by the application.

How close is CryptDB to “optimal” security? Fundamentally, op- timal security is achieved by recent work in theoretical cryptography enabling any computation over encrypted data [18]; however, such proposals are prohibitively impractical. In contrast, CryptDB is prac- tical, and in 8.3, we demonstrate that it also provides significant security in practice. Specifically, we show that all or almost all of the most sensitive fields in the tested applications remain encrypted with highly secure encryption schemes. For such fields, CryptDB provides optimal security, assuming their value is independent of the pattern in which they are accessed (which is the case for medical information, social security numbers, etc). CryptDB is not optimal for fields requiring more revealing encryption schemes, but we find that most such fields are semi-sensitive (such as timestamps).

Finally, we believe that a passive attack model is realistic because malicious DBAs are more likely to read the data, which may be hard to detect, than to change the data or query results, which is more likely to be discovered. In 9, we cite related work on data integrity that could be used in complement with our work. An active adversary that can insert or update data may be able to indirectly compromise confidentiality. For example, an adversary that modifies an email field in the database may be able to trick the application into sending a user’s data to the wrong email address, when the user asks the application to email her a copy of her own data. Such active attacks on the DBMS fall under the second threat model, which we now discuss.

2.2 Threat 2: Arbitrary Threats

We now describe the second threat where the application server, proxy, and DBMS server infrastructures may be compromised arbi- trarily. The approach in threat 1 is insufficient because an adversary can now get access to the keys used to encrypt the entire database.

The solution is to encrypt different data items (e.g., data belong- ing to different users) with different keys. To determine the key that should be used for each data item, developers annotate the ap- plication’s database schema to express finer-grained confidentiality policies. A curious DBA still cannot obtain private data by snooping on the DBMS server (threat 1), and in addition, an adversary who compromises the application server or the proxy can now decrypt only data of currently logged-in users (which are stored in the proxy). Data of currently inactive users would be encrypted with keys not available to the adversary, and would remain confidential.

In this configuration, CryptDB provides strong guarantees in the face of *arbitrary* server-side compromises, including those that gain root access to the application or the proxy. CryptDB leaks at most the data of currently active users for the duration of the compromise, even if the proxy behaves§ in a Byzantine fashion. By “duration of a compromise”, we mean the interval from the start of the compromise until any trace of the compromise has been erased from the system. For a read SQL injection attack, the duration of the compromise spans the attacker’s SQL queries. In the above example of an adversary changing the email address of a user in the database, we consider the system compromised for as long as the attacker’s email address persists in the database.

3 QUERIES OVER ENCRYPTED DATA

This section describes how CryptDB executes SQL queries over encrypted data. The threat model in this section is threat 1 from §2.1. *The* DBMS machines and administrators are not trusted, but the application and the proxy are trusted.

CryptDB enables the DBMS server to execute SQL queries on encrypted data almost as if it were executing the same queries on plaintext data. Existing applications do not need to be changed. The DBMS’s query plan for an encrypted query is typically the same as for the original query, except that the operators comprising the query, such as selections, projections, joins, aggregates, and orderings, are performed on ciphertexts, and use modified operators in some cases. CryptDB's proxy stores a secret master key *MK*, the database schema, and the current encryption layers of all columns. The DBMS server sees an anonymized schema (in which table and col- umn names are replaced by opaque identifiers), encrypted user data, and some auxiliary tables used by CryptDB. CryptDB also equips the server with CryptDB-specific user-defined functions (UDFs) that enable the server to compute on ciphertexts for certain operations. Processing a query in CryptDB involves four steps:

1. The application issues a query, which the proxy intercepts and rewrites: it anonymizes each table and column name, and, using the master key *MK*, encrypts each constant in the query with an encryption scheme best suited for the desired operation (§3.1).
2. The proxy checks if the DBMS server should be given keys to adjust encryption layers before executing the query, and if so, issues an UPDATE query at the DBMS server that invokes a UDF to adjust the encryption layer of the appropriate columns (§3.2).
3. The proxy forwards the encrypted query to the DBMS server, which executes it using standard SQL (occasionally invoking UDFs for aggregation or keyword search).
4. The DBMS server returns the (encrypted) query result, which the proxy decrypts and returns to the application.

3.1

We now describe the encryption types that CryptDB uses, including a number of existing cryptosystems, an optimization of a recent scheme, and a new cryptographic primitive for joins. For each encryption type, we explain the security property that CryptDB requires from it, its functionality, and how it is implemented.

Random (RND). RND provides the maximum security in CryptDB: indistinguishability under an adaptive chosen-plaintext attack (IND-CPA); the scheme is probabilistic, meaning that two equal values are mapped to different ciphertexts with overwhelming probabilityS.On the other hand, RND does not allow any compu- tation to be performed efficiently on the ciphertextQ. An efficient construction of RND is to use a block cipher like AES or Blowfish in CBC Lmode together with a random initialization vector (IV). (We mostly use AES, except for integer values,- where we use Blowfish for its 64-bit block size because the 128-bit block size of AES woulda cause the ciphertext to be significantly longer).

Since, in this threat model,w CryptDB assumes the server does not change results, CryptDB does not require a stronger INDa-CCA2 construction (which would be secure under a chosen-ciphertext attack). However, it wouldr be straightforward to use an IND-CCA2- secure implementation of RND instead, such as a block cipher in UFE mode [13], if needed. e

Deterministic (DET). DET has a slightly weaker guarantee, yet it still provides strong security: it leaks only which encrypted values correspond to the same data value, by Edeterministically generating the same ciphertext for the same plaintext. This encryption layer allows the server to perform equality checks, which means it can perform selects with equality predicates, equality joins, GROUP BY, COUNT, DISTINCT, etc.

In cryptographic terms, DET should be a pseudo-random permu- tation (PRP) [20]. For 64-bit and 128-bit values, we use a block cipher with a matching block size (Blowfish and AES respectively); we make the usual assumption that the AES and Blowfish block ciphers are PRPs. We pad smaller values out to 64 bits, but for data that is longer than a single 128-bit AES block, the standard

CBC mode of operation leaks prefix equality (e.g., if two data items have an identical prefix that is at least 128 bits long). To avoid this problem, we use AES with a variant of the CMC mode [24], which can be approximately thought of as one round of CBC, followed by another round of CBC with the blocks in the reverse order. Since the goal of DET is to reveal equality, we use a zero IV (or "tweak" [24]) for our AES-CMC implementation of DET.

Order-preserving encryption (OPE). OPE allows order rela- tions between data items to be established based on their en- crypted values, without revealing the data itself. If $x < y$, then

$\mathrm{OPE}_K(x) < \mathrm{OPE}_K(y)$, for any secret key K. Therefore, if a column
given encrypted constants OPE (c) and OPE (c) corresponding to the range [$_K c$,$_1$ c]. The $_K$
server$_2$ can also perform ORDER BY$_1$, MIN$_2$, is encrypted with OPE, the
server can perform range queries when
MAX, SORT , etc.

OPE is a weaker encryption scheme than DET because it reveals order. Thus, the CryptDB proxy will only reveal OPE-encrypted columns to the server if users request order queries on those columns. OPE has provable security guarantees [4]: the encryption is equiva- lent to a random mapping that preserves order.

The scheme we use [4] is the first provably secure such scheme. Until CryptDB, there was no implementation nor any measure of the practicality of the scheme. The direct implementation of the scheme took 25 ms per encryption of a 32-bit integer on an Intel 2.8 GHz Q9550 processor. We improved the algorithm by using AVL binary search trees for batch encryption (e.g., database loads), reducing the cost of OPE encryption to 7 ms per encryption without affecting its security. We also implemented a hypergeometric sampler that lies at the core of OPE, porting a Fortran implementation from 1988 [25].

Homomorphic encryption (HOM). HOM is a secure probabilis- tic encryption scheme (IND-CPA secure), allowing the server to perform computations on encrypted data with the final result de- crypted at the proxy. While fully homomorphic encryption is pro- hibitively slow [10], homomorphic encryption for specific operations is efficient. To support summation, we implemented the Paillier cryptosystem [35]. With
Paillier, multiplying the encryptions of HOM (x) HOM (y)
= HOM ($x + y$), where the
multiplication$_K$ is$_K$two values$_K$ results in an encryption of
the sum of the values, i.e., performed$\cdot$ modulo some
public-key value. To compute SUM aggre-
gates, the proxy replaces SUM with calls to a

UDF that performs Paillier multiplication on a column encrypted with HOM. HOM can also be used for computing averages by having the DBMS server return the sum and the count separately, and for incrementing values (e.g., SET *id*=*id*+1), on which we elaborate shortly.

With HOM, the ciphertext is 2048 bits. In theory, it should be possible to pack multiple values from a single row into one HOM $\times$ ciphertext for that row, using the scheme of Ge and Zdonik [17], which would result in an amortized space overhead of 2 (e.g., a 32-bit value occupies 64 bits) for a table with many HOM-encrypted columns. However, we have not implemented this optimization in our prototype. This optimization would also complicate partial- row UPDATEoperations that reset some—but not all—of the values packed into a HOM ciphertext.

Join (JOIN and OPE-JOIN). A separate encryption scheme is necessary to allow equality joins between two columns, because we use different keys for DET to prevent cross-column correlations. JOIN also supports all operations allowed by DET, and also en- ables the server to determine repeating values between two columns. OPE-JOIN enables joins by order relations. We provide a new cryp- tographic scheme for JOIN and we discuss it in §3.4.

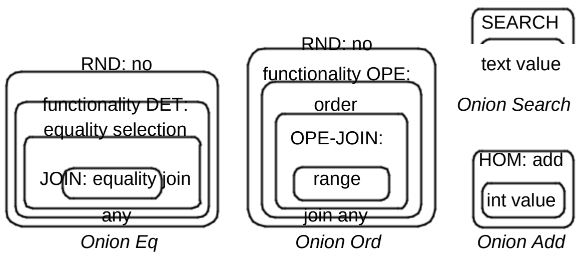


Figure 2: Onion encryption layers and the classes of computation they allow. Onion names stand for the operations they allow at some of their layers (Equality, Order, Search, and Addition). In practice, some onions or onion layers may be omitted, depending on column types or §schema annotations provided by application developers (3.5.2). DET and JOIN are often merged into a§ single onion layer, since§ JOIN is a concatenation of DET and JOIN-ADJ (3.4). A random IV for RND (3.1), shared by the RND layers in *Eq* and *Ord*, is also stored for each data item.

Word search (SEARCH). SEARCH is used to perform searches on encrypted text to support operations such as MySQL's LIKEoper- ator. We implemented the cryptographic protocol of Song et al. [46], which was not previously implemented by the authors; we also use their protocol in a different way, which results in better security guarantees. For each column needing SEARCH, we split the text into keywords using standard delimiters (or using a special keyword extraction function specified by the schema developer). We then remove repetitions in these words, randomly permute the positions of the words, and then encrypt each of the words using Song et al.'s scheme, padding each word to the same size. SEARCH is nearly as secure as RND: the encryption does not reveal to the DBMS server whether a certain word repeats in multiple rows, but it leaks the number of keywords encrypted with SEARCH; an adversary may be able to estimate the number of distinct or duplicate words (e.g., by comparing the size of the SEARCH and RND ciphertexts for the same data).

When the user performs a query such as SELECT * FROM *messages* WHERE *msg* LIKE "% alice %", the proxy gives the DBMS server a token, which is an encryption of alice. The server cannot decrypt the token to figure out the underlying word. Using a user-defined function, the DBMS server checks if any of the word encryptions in any message match the token. In our approach, all the server learns from searching is whether a token matched a mes- sage or not, and this happens only for the tokens requested by the user. The server would learn the same information when returning the result set to the users, so the overall search scheme reveals the minimum amount of additional information needed to return the result.

Note that SEARCH allows CryptDB to only perform full-word keyword searches; it cannot support arbitrary regular expressions. For applications that require searching for multiple adjacent words, CryptDB allows the application developer to disable duplicate re- moval and re-ordering by annotating the schema, even though this is not the default. Based on our trace evaluation, we find that most uses of LIKE can be supported by SEARCH with such schema an-notations. Of course, one can still combine multiple LIKEoperators with AND and OR to check whether multiple independent words are in the text.

3.2

A key part of CryptDB's design is *adjustable query-based encryp- tion*, which dynamically adjusts the layer of encryption on the DBMS server. Our goal is to use the most secure encryption schemes that enable running the requested queries. For example, if the application issues no queries that compare data items in a column, or that sort a

column, the column should**s** be encrypted with RND. For columns that require equality checks but not

inequality checks,**e**DET suf- fices. However, the query set is not always known in
advance. Thus, we need an adaptive scheme

d

that dynamically adjusts encryption strategies.

Our idea is to encrypt each data item in one or more *onions*: that is, each value is dressed in layers of increasingly stronger encryption, as illustrated in Figures 2 and**E** 3. Each layer of each onion enables certain kinds of functionality as explained in the previous**n**subsection. For example, outermost layers such as RND and HOM provide maximum security,**c** whereas inner layers such as OPE provide more functionality.

Multiple onions are needed**r** in practice, both because the compu- tations supported by different encryption schemes**y** are not always strictly ordered, and because of performance

considerations (size of ciphertext**p** and encryption§ time for nested onion layers). Depending on the type of the data

(and any**t**annotations provided by the appli- cation

developer on the database schema, as discussed in**i**3.5.2), CryptDB may not maintain all onions for each column. For instance, the *Search* onion does not make sense for

o

integers, and the *Add* onion does not make sense for strings.

For each layer of each**n**onion, the proxy uses the same key for encrypting values in the same column, and different keys across tables, columns, onions, and onion layers. Using the same key for all values in a column allows the proxy§ to perform operations on a column without having to compute separate keys for each row that will be manipulated. (We use finer-grained encryption keys in 4 to reduce the potential amount of data disclosure in case of an application or proxy server compromise.) Using different keys across columns prevents the server from learning any additional relations. All of these keys are derived from the master key *MK*. For example, for table *t*, column *c*, onion *o*, and encryption layer *l*, the proxy uses the key

$K_{t,c,o,l}$ = PRP$_{MK}$ (table *t,* column *c,* onion *o,* layer *l*), (1) where PRP is a pseudorandom permutation (e.g., AES).

Each onion starts out encrypted with the most secure encryption scheme (RND for onions *Eq* and *Ord*, HOM for onion *Add*, and SEARCH for onion *Search*). As the proxy receives SQL queries from the application, it determines whether layers of encryption need to be removed. Given a predicate *P* on column *c* needed to execute a query on the server, the proxy first establishes what onion layer is needed to compute *P* on *c*. If

§

the encryption of *c* is not already at an onion layer that allows *P*, the proxy strips off the onion layers to allow *P* on *c*, by sending the corresponding onion key to the server. The proxy never decrypts the data past the least-secure encryption onion layer (or past some other threshold layer, if specified by the application developer in the schema, 3.5.1).

CryptDB implements onion layer decryption using UDFs that run on the DBMS server. For example, in Figure 3, to decrypt onion *Ord* of column 2 in table 1 to layer OPE, the proxy issues the following query to the server using the DECRYPT RNDUDF:

```
UPDATE Table1 SET
        C2-Ord = DECRYPT RND(K, C2-Ord, C2-IV)
```

where *K* is the appropriate key computed from Equation (1). At the same time, the proxy updates its own internal state to remember that column *C2-Ord* in *Table1* is now at layer OPE in the DBMS. Each column decryption should be included in a transaction to avoid consistency problems with clients accessing columns being adjusted. Note that onion decryption is performed entirely by the DBMS server. In the steady state, no server-side decryptions are needed, because onion decryption happens only when a new class of com- putation is requested on a column. For example, after an equality

Employees

ID
Name

Table 1

C1-IV	*C1-Eq*	*C1-Ord*	*C1-Add*	*C2-IV*	*C2-Eq*	*C2-Ord*	*C2-Search*
x27c3	x2b82	xcb94	xc2e4	x8a13	xd1e3	x7eb1	

Figure 3: Data layout at the server. When the application creates the table shown on the left, the table created at the DBMS server is the one shown on the right. Ciphertexts shown are not full-length.

check is requested on a column and the server brings the column to layer DET, the column

remains in that state, and future queries with § equality checks require no decryption. This property is the insight into why CryptDB's overhead is modest in the steady state (see 8): the server mostly performs typical SQL processing.

Table1 WHERE *C2-Eq* = xbb..4a, where xbb..4a is the *Eq* onion encryption of "Bob" using $K_{T1,C2,Eq,\text{JOIN}}$ and $K_{T1,C2,Eq,\text{DET}}$.

3.3 E x e c u

Once the onion layers in the DBMS are at the layer necessary to execute**t** a query, the proxy transforms the query to operate on these onions. In particular, the proxy replaces**i** column names in a query with corresponding onion names, based on the class of computation**n** performed on that column. For example, for the schema shown in Figure 3, a reference to the**g** *Name* column for an equality comparison will be replaced with a reference to the *C2-Eq* column.

The proxy also replaces each constant in the query with a corre- sponding**o** onion encryption of that constant, based on the compu- tation in which it is used. For instance, if a query contains WHERE Name = 'Alice', the proxy encrypts**v** 'Alice' by successively ap- plying all encryption layers corresponding to onion *Eq***e**that have not yet been removed from *C2-Eq*.

Finally, the server replaces**r** certain operators with UDF-based counterparts. For instance, the SUM aggregate operator and the + column-addition operator must be replaced with an invocation of a UDF that performs HOM **E**addition of ciphertexts. Equality and order operators (such as = and <) do not need such replacement and can be applied directly to the DET and OPE**n** ciphertexts.

Once the proxy has transformed the query, it sends the query to the DBMS**c** server, receives query results (consisting of encrypted data), decrypts the results using the corresponding onion keys, and sends the decrypted result to the application.

Read query execution. To understand query execution over ci- phertexts, consider the example schema shown in Figure 3. Initially, each column in the table is dressed in all onions of encryption, with RND, HOM, and SEARCH as outermost layers, as shown in Fig- ure 2. At this point, the server can learn nothing about the data other than the number of columns, rows, and data size.

To illustrate when onion layers are removed, consider the query:

SELECT *ID* FROM *Employees* WHERE *Name* = 'Alice',

which requires lowering the encryption of *Name* to layer DET. To execute this query, the proxy first issues the query

UPDATE *Table1* SET
    C2-Eq = DECRYPT R~~N~~D($K_{T1,C2,Eq,\text{RND}}$, *C2-Eq*, *C2-IV*),

where column *C2* corresponds to *Name*. The proxy then issues

SELECT *C1-Eq*, *C1-IV* FROM *Table1* WHERE *C2-Eq* = x7..d,

where column *C1* corresponds to *ID*, and where x7..d is the *Eq* onion encryption of “Alice” with keys $K_{T1,C2,Eq,\text{JOIN}}$ and $K_{T1,C2,Eq,\text{DET}}$ (see Figure 2). Note that the proxy must request the random IV from column C1-IVin order to decrypt the RND ciphertext from C1-Eq. Finally, the proxy decrypts the results from the server using keys $K_{T1,C1,Eq,\text{RND}}$, $K_{T1,C1,Eq,\text{DET}}$, and $K_{T1,C1,Eq,\text{JOIN}}$, obtains the result 23, and returns it to the application.

If the next query is SELECT COUN*T( ) FROM *Employees* WHERE *Name* = 'Bob', no server- side decryptions are necessary, and the proxy * directly issues the query SELECT COUNT( ) FROM

Write query execution. To support INSERT, DELETE, and UPDATEqueries, the proxy applies the same processing to the predi- cates (i.e., the WHEREclause) as for read queries. DELETEqueries re- quire no additional processing. For all INSERT and UPDATEqueries that set the value of a column to a constant, the proxy encrypts each inserted column's value with each onion layer that has not yet been stripped off in that column.

The remaining case is an UPDATEthat sets a column value based on an existing column value, such as *salary*=*salary*+1. Such an update would have to be performed using HOM, to handle addi-tions. However, in doing so, the values in the OPE and DET onions would become stale. In fact, any hypothetical encryption scheme that simultaneously allows addition and direct comparison on the ciphertext is insecure: if a malicious server can compute the order of the items, and can increment the value by one, the server can repeatedly add one to each field homomorphically until it becomes equal to some other value in the same column. This would allow the server to compute the difference between any two values in the database, which is almost equivalent to knowing their values.

There are two approaches to allow updates based on existing column values. If a column is incremented and then only projected (no comparisons are performed on it), the solution is simple: when a query requests the value of this field, the proxy should request the HOM ciphertext from the *Add* onion, instead of ciphertexts from other onions, because the HOM value is up-to-date. For instance, this approach applies to increment queries in TPC-C. If a column is used in comparisons after it is incremented, the solution is to replace the update query with two queries: a SELECT of the old values to be updated, which the proxy increments and encrypts accordingly, followed by an UPDATE setting the new values. This strategy would work well for updates that affect a small number of rows.

Other DBMS features. Most other DBMS mechanisms, such as transactions and indexing, work the same way with CryptDB over encrypted data as they do over plaintext, with no modifications. For transactions, the proxy passes along any BEGIN, COMMIT, and ABORT queries to the DBMS. Since many SQL operators behave differently on NULLs than on non-NULL values, CryptDB exposes NULL values to the DBMS without encryption. CryptDB does not currently support stored procedures, although certain stored procedures could be supported by rewriting their code in the same way that CryptDB's proxy rewrites SQL statements.

The DBMS builds indexes for encrypted data in the same way as for plaintext. Currently, if the application requests an index on a column, the proxy asks the DBMS server to build indexes on that column's DET, JOIN, OPE, or OPE-JOIN onion layers (if they are exposed), but not for RND, HOM, or SEARCH. More efficient index selection algorithms could be investigated.

3.4 Computing Joins

There are two kinds of joins supported by CryptDB: *equi-joins*, in which the join predicate is based on equality, and *range joins*, which involve order checks. To perform an equi-join of two encrypted columns, the columns should be encrypted with the same key so that the server can see matching values between the two columns. At the same time, to provide better privacy, the DBMS server should not be able to join columns for which the application did not request a join, so columns that are never joined should not be encrypted with the same keys.

If the queries that can be issued, or the pairs of columns that can be joined, are known *a priori*, equi-join is easy to support: CryptDB can use the DET encryption scheme with the same key for each group of columns§ that are joined together. 3.5 describes how the proxy learns the columns to be joined in this case. However, the challenging case is when the proxy does not know the set of columns to be joined *a priori*, and hence does not know which columns should be encrypted with matching keys.

To solve this problem, we introduce a new cryptographic primi- tive, JOIN-ADJ (*adjustable join*), which allows the DBMS server to adjust the key of each column at runtime. Intuitively, JOIN-ADJ can be thought of as a keyed cryptographic hash with the additional prop- erty that hashes can be adjusted to change their key *without* § *access to the plaintext*. JOIN-ADJ is a deterministic function of its input, which means that if two plaintexts are equal, the corresponding JOIN-ADJ values are also equal. JOIN-ADJ is collision-resistant, and has a sufficiently long output length (192 bits) to allow us to assume
that collisions never happen in practice. scheme as JOIN(v) =
JOIN-ADJ(v) DET(v),
where denotes con-catenation. This construction allows the proxy to decrypt a JOIN(v)
JOIN-ADJ is non-invertible, so we define the JOIN encryption
column to obtain v by decrypting‖ the DET‖ component, and allows
the DBMS server to check two JOIN values for equality by compar- ing the JOIN-ADJ components.

Each column is initially encrypted at the JOIN layer using a different key, thus preventing any joins between columns. When a query requests a join, the proxy gives the DBMS server an onion key to adjust the JOIN-ADJ values in one of the two columns, so that it matches the JOIN-ADJ key of the other column (denoted the *join-base* column). After the adjustment, the columns share the same JOIN-ADJ key, allowing the DBMS server to join them for equality. The DET components of JOIN remain encrypted with different keys. Note that our adjustable join is transitive: if the user joins columns *A* and *B* and then joins columns *B* and *C*, the server can join *A* and *C*. However, the server cannot join columns in different “transitivity groups”. For instance, if columns *D* and *E* were joined together, the DBMS server would not be able to join columns *A* and *D* on its own. After an initial join query, the JOIN-ADJ values remain trans- formed with the same key, so no re-adjustments are needed for subsequent join queries between the same two columns. One ex- ception is if the application issues another query, joining one of the adjusted columns with a third column, which causes the proxy to re- adjust the column to another join-base.

To avoid oscillations and to converge to a state where all columns in a transitivity group share the same join-base, CryptDB chooses the first column in lexicographic order on table and column name as the join-base. For *n* columns, the overall maximum number of join transitions is $n(n$ $1)/2$.

For range joins, a similar dynamic re-adjustment scheme is diffi-
cult to construct due to lack of structure in OPE schemes. Instead, CryptDB requires that pairs of columns that will be involved in such joins be declared by the application ahead of time, so that matching keys are used for layer OPE-JOIN of those columns; otherwise, the same key will be used for all columns at layer OPE-JOIN. Fortu- nately, range joins are rare; they are not used in any of our example applications, and are used in only 50 out of 128,840 columns in a large SQL query trace we describe in 8, corresponding to just three distinct applications.

tography (ECC). JOIN -ADJ$_K$ (v) is computed as **JOIN-ADJ construction.** Our algorithm uses elliptic-curve cryp-

$$\text{JOIN-ADJ}_K(v) := P^{K \cdot \text{PRF}_{K0}(v)}, \qquad (2)$$

where K is the initial key for that table, column, onion, and layer, P is a point on an elliptic curve (being a public parameter), and PRF$_{K0}$ is a pseudo-random function [20] mapping values to a pseudorandom

number, such as AES$_{K0}$ (SHA(v)), with K_0 being a

key that is the

same for all columns and derived from *MK*. The "exponentiation" is in fact repeated geometric addition of elliptic curve points; it is considerably faster than RSA exponentiation.

at the join l ye , the proxy computes $\Delta K = K/K_J$
(in an appropriate group) and sends$_J$ it to the server.$_J$ Then, given JOIN- ADJ$_K$ (v) (the a query joins columns c and c , each havingWhenkys K and K ^{J}) and ΔK, the JOIN-ADJ values from column c DBMS server uses a $_J$
UDF to adjust the key in c by computing:

$$(\text{JOIN-ADJ}_{K_J}(v))^{\Delta K} = P^{K^J \cdot \text{PRF}_{K0}(v) \cdot (K/K^J)}$$

$$= P^{K \cdot \text{PRF}_{K0}(v)} = \text{JOIN-ADJ}_K(v).$$

Now columns c and c_J share the same JOIN-ADJ key, and the DBMS server can perform an equi-join on c and c_J by taking the JOIN-ADJ component of the JOIN onion ciphertext.

At a high level, the security of this scheme is that the server cannot infer join relations among groups of columns that were not requested by legitimate join queries, and that the scheme does not reveal the plaintext. We proved the security of this scheme based on the standard Elliptic-Curve Decisional Diffie-Hellman hardness as- sumption, and implemented it using a NIST-approved elliptic curve. We plan to publish a more detailed description of this algorithm and the proof on our web site [37].

Improving Security and Performance

Although CryptDB can operate with an unmodified and unannotated schema, as described above, its security and performance can be improved through several optional optimizations, as described below.

3.5.1 S e c u r i t y I m p

Minimum onion layers. Application developers can specify the lowest onion encryption layer that may be revealed to the server for a specific column. In this way, the developer can ensure that the proxy will not execute queries exposing sensitive relations to the server. For example, the developer could specify that credit card numbers should always remain at RND or DET.

In-proxy processing. Although CryptDB can evaluate a number of predicates on the server, evaluating them in the proxy can improve security by not revealing additional information to the server. One common use case is a SELECTquery that sorts on one of the selected columns, without a LIMITon the number of returned columns. Since the proxy receives the entire result set from the server, sorting these results in the proxy does not require a significant amount of compu- tation, and does not increase the bandwidth requirements. Doing so avoids revealing the OPE encryption of that column to the server.

Training mode. CryptDB provides a training mode, which allows a developer to provide a trace of queries and get the resulting onion encryption layers for each field, along with a warning in case some query is not supported. The developer can then examine the resulting encryption levels to understand what each encryption scheme leaks, as described in 2.1. If some onion level is too low for a sensitive field, she should arrange to have the query processed in the proxy (as described above), or to process the data in some other fashion, such as by using a local instance of SQLite.

Onion re-encryption. In cases when an application performs in- frequent queries requiring a low onion layer (e.g., OPE), CryptDB could be extended to re-encrypt onions back to a higher layer after the infrequent query finishes executing. This approach reduces leak- age to

attacks happening in the time window when the
data is at the higher onion layer.

3.5.2 Performance optimizations

Developer annotations. By default, CryptDB encrypts all fields and creates all applicable onions for each data item based on its type. If many columns are not sensitive, the developer can instead provide explicit annotations indicating the sensitive fields (as described in §4), *and* leave the remaining fields in plaintext.

Known query set. If the developer knows some of the queries ahead of time, as is the case for many web applications, the developer can use the training mode described above to adjust onions to the correct layer *a priori*, avoiding the overhead of runtime onion adjust- ments. If the developer provides the exact query set, or annotations that certain functionality is not needed on some columns, CryptDB can also discard onions that are not needed (e.g., discard the *Ord* onion for columns that are not used in range queries, or discard the *Search* onion for columns where keyword search is not performed), discard onion layers that are not needed (e.g., the adjustable JOIN layer, if joins are known *a priori*), or discard the random IV needed for RND for some columns.

§

Ciphertext pre-computing and caching. The proxy spends a sig- nificant amount of time encrypting values used in queries with OPE and HOM. To reduce this cost, the proxy pre-computes (for HOM) and caches (for OPE) encryptions of frequently used constants under different keys. Since HOM is probabilistic, ciphertexts cannot be reused. Therefore, in addition, the proxy pre- computes HOM's Pail- lier r^n randomness values for future encryptions of any data. This optimization reduces the amount of CPU time spent by the proxy on OPE encryption, and assuming the proxy is occasionally idle to perform HOM pre-computation, it removes HOM encryption from the critical path.

4 MULTIPLE PRINCIPALS

We now extend the threat model to the case when the application infrastructure and proxy are also untrusted (threat 2). This model is especially relevant for a multi-user web site running a web and application server. To understand both the problems faced by a multi- user web application and CryptDB's solution to these problems, consider phpBB, a popular online web forum. In phpBB, each user has an account and a password, belongs to certain groups, and can send private messages to other users. Depending on their groups' permissions, users can read entire forums, only forum names, or not be able to read a forum at all.

There are several confidentiality guarantees that would be useful in phpBB. For example, we would like to ensure that a private message sent from one user to another is not visible to anyone else; that posts in a forum are accessible only to users in a group with access to that forum; and that the name of a forum is shown only to users belonging to a group that's allowed to view it. CryptDB provides these guarantees in the face of arbitrary compromises, thereby limiting the damage caused by a compromise.

Achieving these guarantees requires addressing two challenges. First, CryptDB must capture the application's access control policy for shared data at the level of SQL queries. To do this, CryptDB requires developers to annotate their database schema to specify principals and the data that each principal has access to, as described in 4.1.

The second challenge is to reduce the amount of information that an adversary can gain by compromising the system. Our solution limits the leakage resulting from a compromised application or proxy server to just the data accessible to users who were logged in during the compromise. In particular, the attacker cannot access the data of users that were not logged in during the compromise. Leaking the data of active users in case of a compromise is unavoidable: given the impracticality of arbitrary computation on encrypted data, some data for active users must be decrypted by the application.

In CryptDB, each user has a key (e.g., her application-level pass- word) that gives her access to her data. CryptDB encrypts different data items with different keys, and enforces the
§
access control policy using chains of keys starting from user passwords and ending in the encryption keys of SQL data items, as described in 4.2. When a user logs in, she provides her password to the proxy (via the applica- tion). The proxy uses this password to derive onion keys to process queries on encrypted data, as presented in the previous section, and to decrypt the results. The proxy can decrypt only the data that the user has access to, based on the access control policy. The proxy gives the decrypted data to the application, which can now compute on it. When the user logs out, the proxy deletes the user's key.

4.1 Policy Annotations

To express the data privacy policy of a database-backed application at the level of SQL queries, the application developer can annotate the schema of a database in CryptDB by specifying, for any subset of data items, which *principal* has access to it. A principal is an entity, such as a user or a group, over which it is natural to specify an access policy. Each SQL query involving an annotated data item requires the privilege of the corresponding principal. CryptDB defines its own notion of principals instead of using existing DBMS principals for two reasons: first, many applications do not map application-level users to DBMS principals in a sufficiently fine-grained manner, and second, CryptDB requires explicit delegation of privileges between principals that is difficult to extract in an automated way from an access control list specification.

An application developer annotates the schema using the three steps described below and illustrated in Figure 4. In all examples we show, italics indicate table and column names, and bold text indicates annotations added for CryptDB.

Step 1. The developer must define the *principal types* (using PRINCTYPE) used in her application, such as users, groups, or mes- sages. A *principal* is an instance of a principal type, e.g., principal 5 of type user. There are two classes of principals: external and internal. External principals correspond to end users who explicitly authenticate themselves to the application using a password. When a user logs into the application, the application must provide the user password to the proxy so that the user can get the privileges of her external principal. Privileges of other (internal) principals can be acquired only through delegation, as described in Step 3. When the user logs out, the application must inform the proxy, so that the proxy forgets the user's password as well as any keys derived from the user's password.

Step 2. The developer must specify which columns in her SQL schema contain sensitive data, along with the principals that should have access to that data, using the ENC FOR annotation. CryptDB requires that for each private data item in a row, the name of the principal that should have access to that data be stored in another column in the same row. For example, in Figure 4, the decryption of *msgtext* x37a21f is available only to principal 5 of type msg.

Step 3. Programmers can specify rules for how to delegate the privileges of one principal to other principals, using the speaks- for relation [49]. For example, in phpBB, a user should also have the privileges of the groups she belongs to. Since many applica- tions store such information in tables, programmers can specify to CryptDB how to infer delegation rules from rows in an existing table. In particular, programmers can annotate a table *T* with (*a* x) SPEAKS FOR (*b* y). This annotation indicates that each row
present in that table specifies that principal *a* of type x speaks for

```
PRINCTYPE physical user
EXTERNAL; PRINCTYPE user, msg;

CREATE TABLE privmsgs (
  msgid int,
  subject varchar(255) ENC FOR (msgid msg),
  msgtext text         ENC FOR (msgid msg) );

CREATE TABLE privmsgs to ( msgid
  int, rcpt id int, sender id int,
  (sender id user) SPEAKS FOR (msgid
  msg), (rcpt id user) SPEAKS FOR
  (msgid msg) );

CREATE TABLE users (
  userid int, username varchar(255),
  (username physical user) SPEAKS FOR (userid user)
  );
```

Example table contents, without anonymized column

msgid	*subject*
	msgtext

Table *rivmsgs*

msgid	*rcpt id*	*sender id*
5	1	2

Table *users*

userid	
1	‘Alice’
2	‘Bob’

Figure 4: Part of phpBB’s schema with annotations to secure private messages. Only the sender and receiver may see the private message. An attacker that gains complete access to phpBB and the DBMS can access private messages of only currently active users.

principal *b* of type y, meaning that *a* has access to all keys that *b* has access to. Here, x and y must always be fixed principal types. Princi- pal *b* is always specified by the name of a column in table *T* . On the other hand, *a* can be either the name of another column in the same table, a constant, or *T2.col*, meaning *all* principals from column *col* of table *T2*. For example, in Figure 4, principal “Bob” of type physical user speaks for principal 2 of type user, and in Figure 6, all principalsş in the *contactId* column from table *PCMember* (of type contact) speak for the *paperId* principal of type review. Optionally, the programmer can specify a predicate, whose inputs are values in the same row, to specify a condition under which delegation should occur, such as excluding conflicts in Figure 6. 5 provides more examples of using annotations to secure applications.

4.2 Key Chaining

Each principal (i.e., each instance of each principal type) is asso- ciated with a secret, randomly chosen key. If principal *B* speaks for principal *A* (as a result of some SPEAKS FOR annotation), then principal *A*’s key is encrypted using principal *B*’s key, and stored as a row in the special *access keys* table in the database. This allows principal *B* to gain access to principal *A*’s key. For example, in Figure 4, to give users 1 and 2 access to message 5, the key of msg 5 is encrypted with the key of user 1, and also separately encrypted with the key of user 2.

Each sensitive field is encrypted with the key of the principal in the ENC FORannotation. CryptDB encrypts the sensitive field with onions in the same way as for single-principal CryptDB, except that onion keys are derived from a principal’s key as opposed to a global master key.

The key of each principal is a combination of a symmetric key and a public–private key pair. In the common case, CryptDB uses the symmetric key of a principal to encrypt any data and other principals’ keys accessible to this principal, with little CPU cost. However, this is not always possible, if some principal is not currently online. For example, in Figure 4, suppose Bob sends message 5 to Alice, but Alice (user 1) is not online. This means that CryptDB does not have access to user 1's key, so it will not be able to encrypt message 5's key with user 1's symmetric key. In this case, CryptDB looks up the public key of the principal (i.e., user 1) in a second table, *public keys*, and encrypts message 5's key using user 1's public key. When user 1 logs in, she will be able to use the secret key part of her key to decrypt the key for message 5 (and re-encrypt it under her symmetric key for future use).

For external principals (i.e., physical users), CryptDB assigns a random key just as for any other principal. To give an external user access to the corresponding key on login, CryptDB stores the key of each external principal in a third table, *external keys*, encrypted with the principal's password. This allows CryptDB to obtain a user's key given the user's password, and also allows a user to change her password without changing the key of the principal.

When a table with a SPEAKS FOR relation is updated, CryptDB must update the *access keys* table accordingly. To insert a new row into *access keys* for a new SPEAKS FOR relation, the proxy must have access to the key of the principal whose privileges are being delegated. This means that an adversary that breaks into an application or proxy server cannot create new SPEAKS FORrelations for principals that are not logged in, because neither the proxy nor the adversary have access to their keys. If a SPEAKS FORrelation is removed, CryptDB revokes access by removing the corresponding row from *access keys*.

When encrypting data in a query or decrypting data from a result, CryptDB follows key chains starting from passwords of users logged in until it obtains the desired keys. As an optimization, when a user logs in, CryptDB's proxy loads the keys of some principals to which the user has access (in particular, those principal types that do not have too many principal instances—e.g., for groups the user is in, but not for messages the user received).

Applications inform CryptDB of users logging in or out by issuing INSERT and DELETESQL queries to a special table *cryptdb active* that has two columns, *username* and *password*. The proxy intercepts all queries for *cryptdb active*, stores the passwords of logged-in users in memory, and never reveals them to the DBMS server.

CryptDB guards the data of inactive users at the time of an attack. If a compromise occurs, CryptDB provides a bound on the data leaked,

allowing the administrators to not issue§ a blanket warning to *all* the users of the system. In this respect, CryptDB is different from other approaches to database security (see 9). However, some special users such as administrators with access to a large pool of data enable a larger compromise upon an attack. To avoid attacks happening when the administrator is logged in, the administrator should create a separate user account with restricted permissions when accessing the application as a regular user. Also, as good practice, an application should automatically log out users who have been inactive for some period of time.

5 APPLICATION CASE STUDIES

In this section, we explain how CryptDB can be used to secure three existing multi-user web applications. For brevity, we show simplified schemas, omitting irrelevant fields and type specifiers. Overall, we find that once a programmer specifies the principals in the application's schema, and the delegation rules for them us- ing SPEAKSFOR, protecting additional sensitive fields just requires additional ENC FOR annotations.

phpBB is a widely used open source forum with a rich set of access control settings. Users are organized in groups; both users and groups have a variety of access permissions that the application

```
PRINCTYPE physical user EXTERNAL;
PRINCTYPE user, group, forum post, forum
name;

CREATE TABLE users     ( userid int, username varchar(255),
  (username physical user) SPEAKS FOR (userid user) );

CREATE TABLE usergroup ( userid int, groupid
                             int,
(userid user) SPEAKS FOR (groupid group) );

CREATE TABLE aclgroups ( groupid int, forumid int,
optionid int,
  (groupid group) SPEAKS FOR (forumid
          forum post) IF optionid=20,
  (groupid group) SPEAKS FOR (forumid
          forum name) IF optionid=14);

 CREATE TABLE posts        ( postid int, forumid
                             int,
  post text ENC FOR (forumid forum post) );
```

Figure 5: Annotated schema for securing access to posts in phpBB. A user has access to see the content of posts in a forum if any of the groups that the user is part of has such permissions, indicated by *optionid* 20 in the *aclgroups* table for the corresponding *forumid* and *groupid*. Similarly, *optionid* 14 enables users to see the forum's name.

administrator can choose. We already showed how to secure private messages between two users in phpBB in Figure 4. A more detailed case is securing access to posts, as shown in Figure 5. This example shows how to use predicates (e.g., IF *optionid*=...) to imple- ment a conditional speaks-for relation on principals, and also how one column (*forumid*) can be used to represent multiple principals (of different type) with different privileges. There are more ways to gain access to a post, but we omit them here for brevity.

HotCRP is a popular conference review application [27]. A key policy for HotCRP is that PC members cannot see who reviewed their own (or conflicted) papers. Figure 6 shows CryptDB annota- tions for HotCRP's schema to enforce this policy. Today, HotCRP cannot prevent a curious or careless PC chair from logging into the database server and seeing who wrote each review for a paper that she is in conflict with. As a result, conferences often set up a second server to review the chair's papers or use inconvenient out- of- band emails. With CryptDB, a PC chair cannot learn who wrote each review for her paper, even if she breaks into the application or database, since she does not have the decryption key.[1] The reason is that the SQL predicate "NoConflict" checks if a PC member is conflicted with a paper and prevents the proxy from providing access to the PC chair in the key chain. (We assume the PC chair does not modify the application to log the passwords of other PC members to subvert the system.)

grad-apply is a graduate admissions system used by MIT EECS. We annotated its schema to allow an applicant's folder to be accessed only by

```
PRINCTYPE physical user
EXTERNAL; PRINCTYPE contact,
review;

CREATE TABLE ContactInfo ( contactId int,
email varchar(120),
  (email physical user) SPEAKS FOR (contactId contact) );

CREATE TABLE PCMember ( contactId int );
CREATE TABLE PaperConflict ( paperId int,
contactId int ); CREATE TABLE PaperReview (
  paperId        int,
  reviewerId int ENC FOR (paperId review),
  commentsToPC text ENC FOR (paperId
  review), (PCMember.contactId contact)
  SPEAKS FOR                  *                  *
        (paperId review) IF NoConflict(paperId, contactId)
        );
```

the respective applicant and any faculty us- ing (*reviewers.reviewer id* reviewer), meaning all review- ers, SPEAKS FOR (*candidate id* candidate) in table *candi- dates*, and ... SPEAKS FOR (*letter id* letter)in table *let- ters*. The applicant can see all of her folder data except for letters of recommendation. Overall, grad-apply has simple access control and therefore simple annotations.

[1]Fully implementing this policy would require setting up two PC chairs: a main chair, and a backup chair responsible for reviews of the main chair's papers. HotCRP allows the PC chair to impersonate other PC members, so CryptDB annotations would be used to prevent the main chair from gaining access to keys of reviewers assigned to her paper.

Figure 6: Annotated schema for securing reviews in HotCRP. Reviews and the identity of reviewers providing the review will be available only to PC members (table *PCMember* includes PC chairs) who are not conflicted, and PC chairs cannot override this restriction.

6 DISCUSSION

CryptDB's design supports most relational queries and aggregates on standard data types, such as integers and text/varchar types. Addi-tional operations can be added to CryptDB by extending its existing onions, or adding new onions for specific data types (e.g., spatial and multi-dimensional range queries [43]). Alternatively, in some cases, it may be possible to map complex unsupported operation to simpler ones (e.g., extracting the month out of an encrypted date is easier if the date's day, month, and year fields are encrypted separately).

There are certain computations CryptDB cannot support on en- crypted data. For example, it does not support both computation and comparison on the same column, such as WHERE *salary* > *age* 2+10. CryptDB can process a part of this query, but it would also require some processing on the proxy. In CryptDB, such a query should be (1) rewritten into a sub-query that selects a whole column, SELECT *age* 2+10 FROM *. . .*, which CryptDB computes using HOM, and (2) re-encrypted in the proxy, creating a new col- umn (call it *aux*) on the DBMS server consisting of the newly en- crypted values. Finally, the original query with the predicate WHERE *salary* > *aux* should be run. We have not been affected by this limitation in our test applications (TPC-C, phpBB, HotCRP, and grad-apply).

In multi-principal mode, CryptDB cannot perform server-side computations on values encrypted for different principals, even if the application has the authority of all principals in question, be- cause the ciphertexts are encrypted with different keys. For some computations, it may be practical for the proxy to perform the com- putation after decrypting the data, but for others (e.g., large-scale aggregates) this approach may be too expensive. A possible exten- sion to CryptDB to support such queries may be to maintain multiple ciphertexts for such values, encrypted under different keys.

7 IMPLEMENTATION

The CryptDB proxy consists of a C++ library and a Lua module. The C++ library consists of a query parser; a query encryptor/rewriter, which encrypts fields or includes UDFs in the query; and a re-sult decryption module. To allow applications to transparently use CryptDB, we used MySQL proxy [47] and implemented a Lua mod- ule that passes queries and results to and from our C++ module. We implemented our new cryptographic protocols using NTL [44]. Our

*

*

	Databases	Tables	Columns
Complete schema	8,548	177,154	1,244,216
Used in query	1,193	18,162	128,840

Figure 7: Number of databases, tables, and columns on the sql.mit.edu MySQL server, used for trace analysis, indicating the total size of the schema, and the part of the schema seen in queries during the trace period.

CryptDB implementation consists ~ of 18,000 lines ~ of C++ code and 150 lines ~of Lua code, with another 10,000 lines of test code. CryptDB is portable and we have implemented versions for both
Postgres 9.0 and MySQL 5.1. Our initial Postgres-based imple- mentation is described in an earlier technical report [39]. Porting CryptDB to MySQL required changing only 86 lines of code, mostly in the code for connecting to the MySQL server and declaring UDFs. As mentioned earlier, CryptDB does not change the DBMS; we implement all server-side functionality with UDFs and server-side tables. CryptDB's design, and to a large extent our implementation, should work on top of any SQL DBMS that supports UDFs.

8 EXPERIMENTAL EVALUATION

In this section, we evaluate four aspects of CryptDB: the difficulty of modifying an application to run on top of CryptDB, the types of queries and applications CryptDB is able to support, the level of security CryptDB provides, and the performance impact of using CryptDB. For this analysis, we use seven applications as well as a large trace of SQL queries.

We evaluate the effectiveness of our annotations and the needed application changes§ on the three applications we described in 5 (phpBB, HotCRP, and grad-apply), as well as on a TPC-C query mix (a standard workload in the database industry). We then analyze the functionality and security of CryptDB on three more applications, on TPC-C, and on a large trace of SQL queries. The additional three applications are OpenEMR, an electronic medical records applica - tion storing private medical data of patients; the web application of an MIT class (6.02), storing students' grades; and PHP-calendar, storing people's schedules. The large trace of SQL queries comes from a popular MySQL server at MIT, sql.mit.edu. This server is used primarily by web applications running on scripts.mit.edu, a shared web application hosting service operated by MIT's Student Information Processing Board (SIPB). In addition, this SQL server is used by a number of applications that run on other machines and use sql.mit.eduonly to store their data. Our query trace spans about ten days, and includes approximately 126 million queries. Figure 7 summarizes the schema statistics for sql.mit.edu; each database is likely to be a separate instance of some application.

Finally, we evaluate the overall performance of CryptDB on the
phpBB application and on a query mix from TPC-C, and perform a detailed analysis through microbenchmarks.

In the six applications (not counting TPC-C), we only encrypt sen- sitive columns, according to a manual inspection. Some fields were clearly sensitive (e.g., grades, private message, medical information), but others were only marginally so (e.g., the time when a message was posted). There was no clear threshold between sensitive or not, but it was clear to us which fields were definitely sensitive. In the case of TPC-C, we encrypt all the columns in the database in single-principal mode so that we can study the performance and functionality of a fully encrypted DBMS. All fields are considered for encryption in the large query trace as well.

8.1 Application Changes

Figure 8 summarizes the amount of programmer effort required to use CryptDB in three multi-user web applications and in the single-principal TPC-C queries. The results show that, for multi-principal mode, CryptDB required between 11 and 13 unique schema annota- tions (29 to 111 in total), and 2 to 7 lines of code changes to provide user passwords to the proxy, in order to secure sensitive information stored in the database. Part of the simplicity is because securing an additional column requires just one annotation in most cases. For the single-principal TPC-C queries, using CryptDB required no application annotations at all.

8.2 Functional Evaluation

To evaluate what columns, operations, and queries CryptDB can support, we analyzed the queries issued by six web applications (including the three applications we analyzed in 8.1), the TPC-C queries, and the SQL queries from sql.mit.edu. The results are shown in the left half of Figure 9.

CryptDB supports most queries; the number of columns in the "needs plaintext" column, which counts columns that cannot be processed in encrypted form by CryptDB, is small relative to the total number of columns. For PHP-calendar and OpenEMR, CryptDB does not support queries on certain sensitive fields that perform string manipulation (e.g., substring and lowercase conversions) or date manipulation (e.g., obtaining the day, month, or year of an encrypted date). However, if these functions were precomputed with the result added as standalone columns (e.g., each of the three parts of a date were encrypted separately), CryptDB would support these queries.

The next two columns, "needs HOM" and "needs SEARCH", reflect the number of columns for which that encryption scheme is needed to process some queries. The numbers suggest that these encryption schemes are important; without these schemes, CryptDB would be unable to support those queries.

Based on an analysis of the larger sql.mit.edu trace, we found that CryptDB should be able to support operations over all but 1,094 of the 128,840 columns observed in the trace. The "in-proxy processing" shows analysis results where we assumed the proxy can perform some lightweight operations on the results returned from the DBMS server. Specifically, this included any operations that are not needed to compute the set of resulting rows or to aggregate rows (that is, expressions that do not appear in a WHERE, HAVING, or GROUP BY clause, or in an ORDER BY clause with a LIMIT, and are not aggregate operators). With in-proxy processing, CryptDB should be able to process queries over encrypted data over all but 571 of the 128,840 columns, thus supporting 99.5% of the columns.

Of those 571 columns, 222 use a bitwise operator in a WHERE clause or perform bitwise aggregation, such as the Gallery2 applica- tion, which uses a bitmask of permission fields and consults them in WHERE clauses. Rewriting the application to store the permissions in a different way would allow CryptDB to support such opera- tions. Another 205 columns perform string processing in the WHERE clause, such as comparing whether lowercase versions of two strings match. Storing a keyed hash of the lowercase version of each string for such columns, similar to the JOIN-ADJ scheme, could support case-insensitive§ equality checks for ciphertexts. 76 columns are involved in mathematical transformations in the WHERE clause, such as manipulating dates, times, scores, and geometric coordinates. 41 columns invoke the LIKE operator with a column reference for the pattern; this is typically used to check a particular value against a table storing a list of banned IP addresses, usernames, URLs, etc. Such a query can also be rewritten if the data items are sensitive.

8.3 Security Evaluation

To understand the amount of information that would be revealed to the adversary in practice, we examine the steady-state onion levels of different columns for a range of applications and queries. To

Applicatio	Annotations	Login/logout code	Sensitive fields secured, and examples of such fields
n phpBB	31 (11 unique)	7 lines	23: private messages (content, subject), posts,
HotCRP	forums		
grad-apply	29 (12 unique)	2 lines	22: paper content and paper information, reviews
TPC-C (single	111 (13 unique)	2 lines	103: student grades (61), scores (17),
princ.)	recommendations, reviews		
	0	0	92: all the fields in all the tables encrypted

Figure 8: Number of annotations the programmer needs to add to secure sensitive fields, lines of code to be added to provide CryptDB with the passwords of users, and the number of sensitive fields that CryptDB secures with these annotations, for three different applications. We count as one annotation each invocation of our three types of annotations and any SQL predicate used in a SPEAKS FOR annotation. Since multiple fields in the same table are usually encrypted for the same principal (e.g., message subject and content), we also report unique annotations.

Application	**Total cols.**	**Consider for enc.**	**Needs plaintext**	**Needs HOM**	**Needs SEARCH**	**Non-plaintext cols. with MinEnc:**				**Most sensitive**
						RND	**SEARCH**	**DET**	**OPE**	**cols. at HIGH**
phpBB	563	23	0	1	0	21	0	1	1	6 / 6
HotCRP	204	22	0	2	1	18	1	1	2	18 / 18
grad-apply	706	103	0	0	2	95	0	6	2	94 / 94
OpenEMR	1, 297	566	7	0	3	526	2	12	19	525 / 540
MIT 6.02	15	13	0	0	0	7	0	4	2	1 / 1
PHP-calendar	25	12	2	0	2	3	2	4	1	3 / 4
TPC-C	92	92	0	8	0	65	0	19	8	—
Trace from sql.mit.edu	128, 840	128, 840	1, 094	1, 019	1, 125	80, 053	350	34, 212	13, 131	—
. . . with in-proxy processing	128, 840	128, 840	571	1, 016	1, 135	84, 008	398	35, 350	8, 513	—
. . . col. name contains *pass*	2, 029	2, 029	2	0	0	1, 936	0	91	0	—
. . . col. name contains *content*	2, 521	2, 521	0	0	52	2, 215	52	251	3	—
. . . col. name contains *priv*	173	173	0	4	0	159	0	12	2	—

Figure 9: Steady-state onion levels for database columns required by a range of applications and traces. “Needs plaintext” indicates that CryptDB cannot execute the application’s queries over encrypted data for that column. For the applications in the top gr oup of rows, sensitive columns were determined manually, and only these columns were considered for encryption. For the bottom group of rows, all database columns were automatically considered for encryption. The rightmost column considers the application’s most sensitive database columns, and reports the number of them that have MinEnc in HIGH (both terms are defined in §8.3).

quantify the level of security, we define the MinEnc of a column to be the weakest onion encryption scheme exposed on any of the onions of a column when onions reach a steady state (i.e., after the application generates all query types, or after running the whole trace). We consider RND and HOM to be the strongest schemes, followed by SEARCH, followed by DET and JOIN, and finishing with the weakest scheme which is OPE. For example, if a column has onion *Eq* at RND, onion *Ord* at OPE and onion *Add* at HOM, the MinEnc of this column is OPE.

The right side of Figure 9 shows the MinEnc onion level for a range of applications and query traces. We see that most fields remain at RND, which is the most secure scheme. For example, OpenEMR has hundreds of sensitive fields describing the medical conditions and history of patients, but these fields are mostly just inserted and fetched, and are not used in any computation. A num- ber of fields also remain at DET, typically to perform key lookups and joins. OPE, which leaks order, is used the least frequently, and mostly for fields that are marginally sensitive (e.g., timestamps and counts of messages). Thus, CryptDB’s adjustable security pro- vides a significant improvement in confidentiality over revealing all encryption schemes to the server.

To analyze CryptDB’s security for specific columns that are par- ticularly sensitive, we define a new security level, HIGH, which includes the RND and HOM encryption schemes, as well as DET for columns having no repetitions (in which case DET is logically equivalent to RND). These are highly secure encryption schemes leaking virtually nothing about the data. DET for columns with repeats and OPE are not part of HIGH as they reveal relations to the DBMS server. The rightmost column in Figure 9 shows that most of the particularly sensitive columns (again, according to manual inspection) are at HIGH.

For the sql.mit.edu trace queries, approximately 6.6% of columns were at OPE even with in-proxy processing; other en- crypted columns (93%)

remain at DET or above. Out of the columns that were at OPE, 3.9% are used in an ORDER BY clause with a LIMIT, 3.7% are used in an inequality comparison in a WHEREclause, and 0.25% are used in a MINor MAXaggregate operator (some of the columns are counted in more than one of these groups). It would be difficult to perform these computations in the proxy without substantially increasing the amount of data sent to it.

Although we could not examine the schemas of applications us- ing sql.mit.edu to determine what fields are sensitive—mostly due to its large scale—we measured the same statistics as above for columns whose names are indicative of sensitive data. In particular, the last three rows of Figure 9 show columns whose name contains the word “pass” (which are almost all some type of password), “con- tent” (which are typically bulk data managed by an application), and “priv” (which are typically some type of private message). CryptDB reveals much less information about these columns than an average column, almost all of them are supported, and almost all are at RND or DET.

Finally, we empirically validated CryptDB’s confidentiality guar- antees by trying real attacks on phpBB that have been listed in the CVE database [32], including two SQL injection attacks (CVE-2009- 3052 & CVE-2008-6314), bugs in permission checks (CVE-2010- 1627 & CVE-2008-7143), and a bug in remote PHP file inclusion (CVE-2008-6377). We found that, for users not currently logged in, the answers returned from the DBMS were encrypted; even with root access to the application server, proxy, and DBMS, the answers were not decryptable.

8.4 Performance Evaluation

To evaluate the performance of CryptDB, we used a machine with two 2.4 GHz Intel Xeon E5620 4-core processors and 12 GB of RAM to run the MySQL 5.1.54 server, and a machine with eight 2.4 GHz AMD Opteron 8431 6-core processors and 64 GB of RAM to run the CryptDB proxy and the clients. The two machines were connected over a shared Gigabit Ethernet network. The higher-provisioned client machine ensures that the clients are not the bottleneck in any experiment. All workloads fit in the server’s RAM.

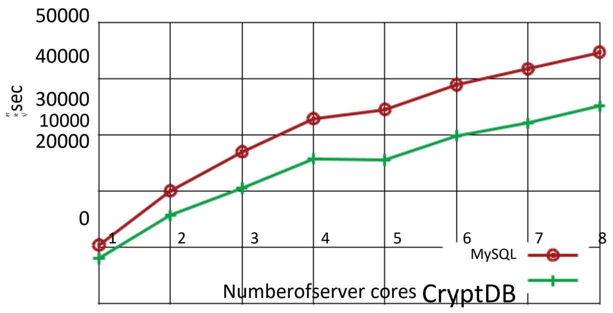


Figure 10: Throughput for TPC-C queries, for a varying number of cores on the underlying MySQL DBMS server.

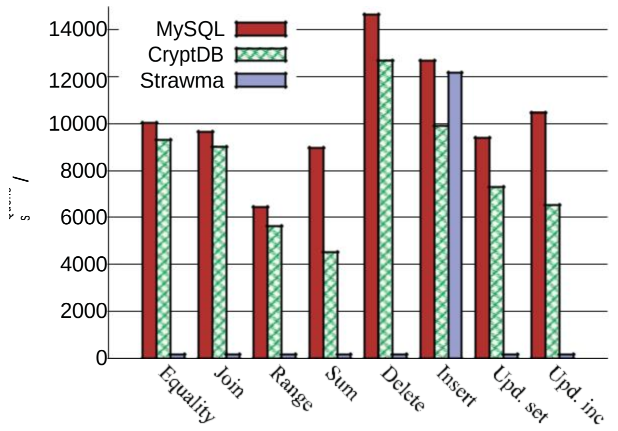


Figure 11: Throughput of different types of SQL queries from the TPC- C query mix running under MySQL, CryptDB, and the strawman design. “Upd. inc” stands for UPDATE that increments a column, and “Upd. set” stands for UPDATE which sets columns to a constant.

8.4.1 *TPC-C*

We compare the performance of a TPC-C query mix when running on an unmodified MySQL server versus on a CryptDB proxy in front of the§ MySQL server. We trained CryptDB on the query set (3.5.2) so there are no onion adjustments during the TPC-C experiments. Figure 10 shows the throughput of TPC-C queries as the number of cores on the server varies from one to eight. In all cases, the server spends 100% of its CPU time processing queries. Both MySQL and CryptDB scale well initially, but start to level off due to internal lock contention in the MySQL server, as reported by SHOW STATUS LIKE ’Table%’. The overall throughput with CryptDB is 21 –26% lower than MySQL, depending on the exact number of cores.

To understand the sources of CryptDB’s overhead, we measure the server throughput for different types of SQL queries seen in TPC-C, on the same server, but running with only one core enabled. Figure 11 shows the results for MySQL, CryptDB, and a *strawman* design; the strawman performs each query over data encrypted with RND by decrypting the relevant data using a UDF, performing the query over× the plaintext, and

×

Query (& scheme)	MySQL Server	CryptDB		
		Server	Proxy	Proxy×
Select by = (DET)	0.10 ms	0.11 ms	0.86	0.86
Select join (JOIN)	0.10 ms	0.11 ms	ms	ms
Select range	0.16 ms	0.22 ms	0.75	0.75
(OPE) Select sum	0.11 ms	0.46 ms	ms	ms
(HOM) Delete	0.07 ms	0.08 ms	**0.78**	28.7
Insert (all)	0.08 ms	0.10 ms	ms	ms
Update set (all)	0.11 ms	0.14 ms	0.99	0.99
Update inc	0 .10 ms	0.17 ms	ms	ms

Figure 12: Server and proxy latency for different types of SQL queries from TPC-C. For each query type, we show the predominant(HOM) encryption scheme used at the0.28server. 0Due.28 to details of the TPC-C workload, each query type affects a different number of rows, and involves a different

re-encrypting the result (if updating rows). The results show that CryptDB’s throughput penalty is great- est for queries that involve a SUM (2.0 less throughput) and for incrementing UPDATE statements (1.6 less throughput); these are the queries that involve HOM additions at the server. For the other types of queries, which form a larger part of the TPC-C mix, the throughput overhead is modest. The strawman design performs poorly for almost all queries because the DBMS’s indexes on the number of cryptographic operations. The left two columns correspond to server throughput, which is also shown in Figure 11. “Proxy” shows the latency added by CryptDB’s proxy; “Proxy×” shows the proxy latency without the ciphertext pre-computing and caching optimization (3.5). Bold numbers show where pre-computing and caching ciphertexts helps. The “Overall” row is the average latency over the mix of TPC-C queries. “Update set” is an UPDATE where the fields are set to a constant, and “Update inc” is an UPDATE where some fields are incremented.

Scheme	Encrypt	Decrypt	Special operation
Blowfish (1 int.)	0.0001 ms	0.0001 ms	—
AES-CBC (1 KB)	0.008 ms	0.007 ms	—
AES-CMC (1 KB)	0.016 ms	0.015 ms	—
OPE (1 int.)	9.0 ms	9.0 ms	Compare: 0 ms
SEARCH (1 word)	0.01 ms	0.004 ms	Match: 0.001 ms
HOM (1 int.)	9.7 ms	0.7 ms	Add: 0.005 ms
JOIN-ADJ (1 int.)	0.52 ms	—	Adjust: 0.56 ms

Figure 13: Microbenchmarks of cryptographic schemes, per unit of data encrypted (one 32-bit integer, 1 KB, or one 15-byte word of text), measured by taking the average time over many iterations.

RND-encrypted data are useless for operations on the underlying plaintext data. It is pleasantly surprising that the higher security of CryptDB over the strawman also brings better performance.

To understand the latency introduced by CryptDB’s proxy, we measure the server and proxy processing times for the same types of SQL queries as above. Figure 12 shows the results. We can see that there is an overall server latency increase of 20% with CryptDB, which we consider modest. The proxy adds an average of 0.60 ms to a query; of that time, 24% is spent in MySQL proxy, 23% is spent in encryption and decryption, and the remaining 53% is spent parsing and processing queries. § The cryptographic overhead is relatively small

because most of our encryption schemes are efficient; Figure 13 shows their performance. OPE and HOM are the slowest, but the ciphertext pre-computing and caching optimization (3.5) masks the high latency of queries requiring OPE and HOM. Proxy× in Figure 12 shows the latency without these optimizations, which is significantly higher for the corresponding query types. SELECT queries that involve a SUM use HOM but do not benefit from this optimization, because the proxy performs decryption, rather than encryption.

In all TPC-C experiments, the proxy used less than 20 MB of memory. Caching ciphertexts for the 30, 000 most common values for OPE accounts for about 3 MB, and pre-computing ciphertexts and randomness for 30,000 values at HOM required 10 MB.

8.4.2 *Mult*

i-User Web Applications §

To evaluate the impact of CryptDB on application performance, we measure the throughput of phpBB for a workload with 10 parallel clients, which ensured 100% CPU load at the server. Each client continuously issued HTTP requests to browse the forum, write and

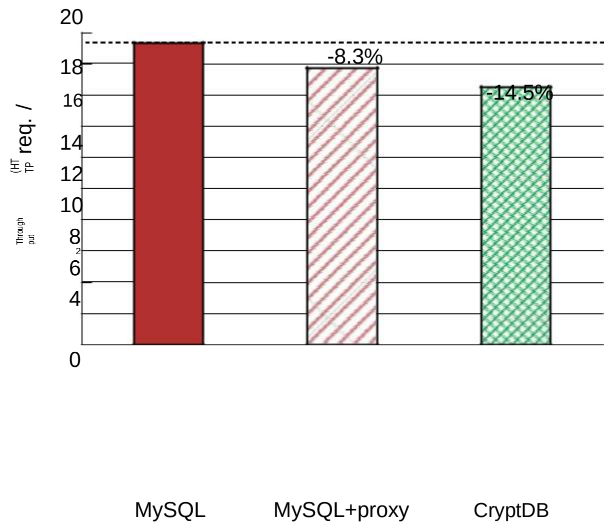


Figure 14: Throughput comparison for phpBB. “MySQL” denotes phpBB running directly on MySQL. “MySQL+proxy” denotes phpBB running on an unencrypted MySQL database but going through MySQL proxy. “CryptDB” denotes phpBB running on CryptDB with notably sensitive fields annotated and the database appropriately encrypted. Most HTTP requests involved tens of SQL queries each. Percentages indicate throughput reduction relative to MySQL. ×

DB	Login	R post	W post	R msg	W msg
MySQL	60 ms	50 ms	133 ms	61 ms	237 ms
CryptDB	67 ms	60 ms	151 ms	73 ms	251 ms

Figure 15: Latency for HTTP requests that heavily use encrypted fields in phpBB for MySQL and CryptDB. R and W stand for read and write.

×

read posts, as well as write and read private messages. We pre-loaded forums and user mailboxes with messages. In this experiment, we co-located the MySQL DBMS, the CryptDB proxy, and the web application server on a single-core machine, to ensure we do not add additional resources for a separate proxy server machine to the system in the CryptDB configuration. In practice, an administrator would likely run the CryptDB proxy on another machine for security. Figure 14 shows the throughput of phpBB in three different con- figurations: (1) connecting to a stock MySQL server, (2) connecting to a stock MySQL server through MySQL proxy, and (3) connecting to CryptDB, with notably sensitive fields encrypted as summarized in Figure 9, which in turn uses a stock MySQL server to store encrypted data. The results show that phpBB incurs an overall throughput loss of just 14.5%, and that about half of this loss comes from inefficiencies in MySQL proxy unrelated to CryptDB. Fig- ure 15 further shows the end-to-end latency for five types of phpBB requests. The results show that CryptDB adds 7–18 ms (6–20%) of processing time per request.

secure onion layers, such as RND, is fast, and needs to be performed only once per column for the lifetime of the system.[2] Removing a layer of RND requires AES decryption, which our experimental machine can perform at 200 MB/s per core. Thus, removing an onion layer is bottlenecked by the speed at which the ˜DBMS server can copy a column from disk for disk-bound databases.

8.4.3 Storage

CryptDB increases the amount of the data stored in the DBMS, because it stores multiple onions for the same field, and because ciphertexts are larger than plaintexts for some encryption schemes. For TPC-C, CryptDB increased the database size by 3.76 , mostly due to cryptographic expansion of integer fields encrypted with HOM (which expand from 32 bits to 2048 bits); strings and binary data remains roughly the same size. For phpBB, the database size using an unencrypted system was 2.6 MB for a workload of about 1,000 private messages and 1,000 forum posts generated by 10 users. The same workload on CryptDB had a database of 3.3 MB, about 1.2 larger. Of the 0.7 MB increase, 230 KB is for storage of *access keys*, 276 KB is for *public keys* and *external keys*, and 166 KB is due to expansion of encrypted fields.

8.4.4 Adjustable Encryption

9 Related Work

Search and queries over encrypted data. Song et al. [46] describe cryptographic tools for performing keyword search over encrypted data, which we use to implement SEARCH. Amanatidis et al. [2]

propose methods for exact searches that do not require scanning the entire database and could be used to process certain restricted SQL queries. Bao et al. [3] extend these encrypted search methods to the multi-user case. Yang et al. [51] run selections with equality predicates over encrypted data. Evdokimov and Guenther present methods for the same selections, as well as Cartesian products and projections [15]. Agrawal et al. develop a statistical encoding that preserves the order of numerical data in a column [1], but it does not have sound cryptographic properties, unlike the scheme we use [4]. Boneh and Waters show public-key schemes for comparisons, subset checks, and conjunctions of such queries over encrypted data [5], but these schemes have ciphertext lengths that are exponential in the length of the plaintext, limiting their practical applicability.

When applied to processing SQL on encrypted data, these tech- niques suffer from some of the following limitations: certain basic queries are not supported or are too inefficient (especially joins and order checks), they require significant client-side query processing, users either have to build and maintain indexes on the data at the server or to perform sequential scans for every selection/search, and implementing these techniques requires unattractive changes to the innards of the DBMS.

Some researchers have developed prototype systems for subsets of SQL, but they provide no confidentiality guarantees, require a significant DBMS rewrite, and rely on client-side processing [9, 12, 22]. For example, Hacigumus et al. [22] heuristically split the domain of possible values
Adjustable query-based encryption involves decrypting columns to

lower-security onion levels. Fortunately, decryption for the more-data/columns.

for each column into partitions, storing the partition number unencrypted for each data item, and rely on extensive client-side filtering of query results. Chow et al. [8] require trusted entities and two non-colluding untrusted DBMSes.

Untrusted servers. SUNDR [28] uses cryptography to provide privacy and integrity in a file system on top of an untrusted file server. Using a SUNDR-like model, SPORC [16] and Depot [30] show how to build low-latency applications, running mostly on the clients, without having to trust a server. However, existing server-side appli-cations that involve separate database and application servers cannot be used with these systems unless they are rewritten as distributed client-side applications to work with SPORC or Depot. Many appli- cations are not amenable to such a structure.

Companies like Navajo Systems and Ciphercloud provide a trusted application-level proxy that intercepts network traffic be- tween clients and cloud-hosted servers (e.g., IMAP), and encrypts sensitive data stored on the server. These products appear to break up sensitive data (specified by application-specific rules) into tokens (such as words in a string), and encrypt each of these tokens using an order-preserving encryption scheme, which allows token-level searching and sorting. In contrast, CryptDB supports a richer set of operations (most of SQL), reveals only relations for the necessary classes of computation to the server based on the queries issued by the application, and allows chaining of encryption keys to user passwords, to restrict data leaks from a compromised proxy.

[2]Unless the administrator periodically re-encrypts

Disk encryption. Various commercial database products, such as Oracle's Transparent Data Encryption [34], encrypt data on disk, but decrypt it to perform query processing. As a result, the server must have access to decryption keys, and an adversary compromising the DBMS software can gain access to the entire data.

Software security. Many tools help programmers either find or mitigate mistakes in their code that may lead to vulnerabilities, including static analysis tools like PQL [29, 31] and UrFlow [7], and runtime tools like Resin [52] and CLAMP [36]. In contrast, CryptDB provides confidentiality guarantees for user data even if the adversary gains complete control over the application and database servers. These tools provide no guarantees in the face of this threat, but in contrast, CryptDB cannot provide confidentiality in the face of vulnerabilities that trick the user's client machine into issuing unwanted requests (such as cross-site scripting or cross-site request forgery vulnerabilities in web applications). As a result, using CryptDB together with these tools should improve overall application security.

Rizvi et al. [41] and Chlipala [7] specify and enforce an applica- tion's security policy over SQL views. CryptDB's SQL annotations can capture most of these policies, except for result processing being done in the policy's view, such as allowing a user to view only aggregates of certain data. Unlike prior systems, CryptDB enforces SQL-level policies cryptographically, without relying on compile-time or run-time permission checks.

Privacy-preserving aggregates. Privacy-preserving data inte- gration, mining, and aggregation schemes are useful [26, 50], but are not usable by many applications because they support only spe- cialized query types and require a rewrite of the DBMS. Differential privacy [14] is complementary to CryptDB; it allows a trusted server to decide what answers to release and how to obfuscate answers to aggregation queries to avoid leaking information about any specific record in the database.

Query integrity. Techniques for SQL query integrity can be integrated into CryptDB because CryptDB allows relational queries on encrypted data to be processed just like on plaintext. These methods can provide integrity by adding a MAC to each tuple [28, 42], freshness using hash chains [38, 42], and both freshness and completeness of query results [33]. In addition, the client can verify the results of aggregation queries [48], and provide query assurance for most read queries [45].

Outsourced databases. Curino et al. advocate the idea of a relational cloud [11], a context in which CryptDB fits well.

10 Conclusion

We presented CryptDB, a system that provides a practical and strong level of confidentiality in the face of two significant threats con- fronting database-backed applications: curious DBAs and arbitrary compromises of the application server and the DBMS. CryptDB meets its goals using three ideas: running queries efficiently over encrypted data using a novel SQL-aware encryption strategy, dy- namically adjusting the encryption level using onions of encryption to minimize the information revealed to the untrusted DBMS server, and chaining encryption keys to user passwords in a way that allows only authorized users to gain access to encrypted data.

Our evaluation on a large trace of 126 million SQL queries from a production MySQL server shows that CryptDB can support opera- tions over encrypted data for 99.5% of the 128,840 columns seen in the trace. The throughput penalty of CryptDB is modest, resulting in a reduction of 14.5–26% on two applications as compared to unmod- ified MySQL. Our security analysis shows that CryptDB protects most sensitive fields with highly secure encryption schemes for six applications. The developer effort consists of 11–13 unique schema annotations and 2–7 lines of source code changes to express relevant privacy policies for 22–103 sensitive fields in three multi-user web applications.

The source code for our implementation is available for download at http://css.csail.mit.edu/cryptdb/.

ACKNOWLEDGMENTS

We thank Martin Abadi, Brad Chen, Carlo Curino, Craig Harris, Evan Jones, Frans Kaashoek, Sam Madden, Mike Stonebraker, Mike Walfish, the anonymous reviewers, and our shepherd, Adrian Perrig, for their feedback. Eugene Wu and Alvin Cheung also provided useful advice. We also thank Geoffrey Thomas, Quentin Smith, Mitch Berger, and the rest of the scripts.mit.edu maintainers for providing us with SQL query traces. This work was supported by the NSF (CNS-0716273 and IIS-1065219) and by Google.

REFERENCES

[1]R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, Paris, France, June 2004.

[2]G. Amanatidis, A. Boldyreva, and A. O'Neill. Provably-secure schemes for basic query support in outsourced databases. In *Pro- ceedings of the 21st Annual IFIP WG 11.3 Working Conference on Database and Applications Security*, Redondo Beach, CA, July 2007.

[3]F. Bao, R. H. Deng, X. Ding, and Y. Yang. Private query on encrypted data in multi-user settings. In *Proceedings of the 4th International Conference on Information Security Practice and Experience*, Sydney, Australia, April 2008.

[4]A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving symmetric encryption. In *Proceedings of the 28th Annual International Conference on the Theory and Applica- tions of Cryptographic Techniques (EUROCRYPT)*, Cologne, Germany, April 2009.

[5]D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *Proceedings of the 4th Conference on Theory of Cryptography*, 2007.

[6]A. Chen. GCreep: Google engineer stalked teens, spied on chats.
Gawker, September 2010. http://gawker.com/5637234/.

[7]A. Chlipala. Static checking of dynamically-varying security
policies in database-backed applications. In *Proceedings of the 9th Symposium on Operating Systems Design and Implementa- tion*, Vancouver, Canada, October 2010.

[8]S. S. M. Chow, J.-H. Lee, and L. Subramanian. Two-party com- putation model for privacy-preserving queries over distributed databases. In *Proceedings of the 16th Network and Distributed System Security Symposium*, February 2009.

[9]V. Ciriani, S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Para- boschi, and P. Samarati. Keep a few: Outsourcing data while maintaining confidentiality. In *Proceedings of the 14th Euro- pean Symposium on Research in Computer Security*, September 2009.

[10]M. Cooney. IBM touts encryption innovation; new technology performs calculations on encrypted data without decrypting it. *Computer World*, June 2009.

[11]C. Curino, E. P. C. Jones, R. A. Popa, N. Malviya, E. Wu, S. Madden, H. Balakrishnan, and N. Zeldovich. Relational cloud: A database-as-a-service for the cloud. In *Proceedings of the 5th Biennial Conference on Innovative Data Systems Re- search*, pages 235–241, Pacific Grove, CA, January 2011.

[12]E. Damiani, S. D. C. di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Balancing confidentiality and efficiency in un- trusted relational DBMSs. In *Proceedings of the 10th ACM Con- ference on Computer and Communications Security*, Washing- ton, DC, October 2003.

[13]A. Desai. New paradigms for constructing symmetric encryp- tion schemes secure against chosen-ciphertext attack. In *Pro- ceedings of the 20th Annual International Conference on Ad- vances in Cryptology*, pages 394–412, August 2000.
[14]C. Dwork. Differential privacy: a survey of results. In *Proceed- ings of the 5th International Conference on Theory and Applica- tions of Models of Computation*, Xi'an, China, April 2008.
[15]S. Evdokimov and O. Guenther. Encryption techniques for se- cure database outsourcing. Cryptology ePrint Archive, Report 2007/335.
[16]A. J. Feldman, W. P. Zeller, M. J. Freedman, and E. W. Felten. SPORC: Group collaboration using untrusted cloud resources. In *Proceedings of the 9th Symposium on Operating Systems De- sign and Implementation*, Vancouver, Canada, October 2010.
[17]T. Ge and S. Zdonik. Answering aggregation queries in a secure system model. In *Proceedings of the 33rd International Con- ference on Very Large Data Bases*, Vienna, Austria, September 2007.
[18]R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology (CRYPTO)*, Santa Barbara, CA, August 2010.
[19]C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, Bethesda, MD, May–June 2009.

[20]O. Goldreich. *Foundations of Cryptography: Volume I Basic Tools*. Cambridge University Press, 2001.
[21]A. Greenberg. DARPA will spend 20 million to search for crypto's holy grail. *Forbes*, April 2011.
[22]H. Hacigumus, B. Iyer, C. Li, and S. Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *Proceedings of the 2002 ACM SIGMOD International Confer- ence on Management of Data*, Madison, WI, June 2002.
[23]J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson,
W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and
E. W. Felten. Lest we remember: Cold boot attacks on encryp- tion keys. In *Proceedings of the 17th Usenix Security Sympo- sium*, San Jose, CA, July–August 2008.
[24]S. Halevi and P. Rogaway. A tweakable enciphering mode. In
Advances in Cryptology (CRYPTO), 2003.
[25]V. Kachitvichyanukul and B. W. Schmeiser. Algorithm 668: H2PEC: Sampling from the hypergeometric distribution. *ACM Transactions on Mathematical Software*, 14(4):397–398, 1988.
[26]M. Kantarcioglu and C. Clifton. Security issues in querying encrypted data. In *Proceedings of the 19th Annual IFIP
WG 11.3 Working Conference on Database and Applications Secu- rity*, Storrs, CT, August 2005.
[27]E. Kohler. Hot crap! In *Proceedings of the Workshop on Or- ganizing Workshops, Conferences, and Symposia for Computer Systems*, San Francisco, CA, April 2008.
[28]J. Li, M. Krohn, D. Mazi e`res, and D. Shasha. Secure untrusted data repository (SUNDR). In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation*, pages 91– 106, San Francisco, CA, December 2004.
[29]V. B. Livshits and M. S. Lam. Finding security vulnerabilities in Java applications with static analysis. In *Proceedings of the 14th Usenix Security Symposium*, pages 271–286, Baltimore, MD, August 2005.
[30]P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin, and M. Walfish. Depot: Cloud storage with minimal trust. In *Proceedings of the 9th Symposium on Operating Systems Design and Implementation*, Vancouver, Canada, October 2010.
[31]M. Martin, B. Livshits, and M. Lam. Finding application er- rors and security flaws using PQL: a program query language. In *Proceedings of the 2005 Conference on Object-Oriented Pro- gramming, Systems, Languages and Applications*, pages 365– 383, San Diego, CA, October 2005.
[32]National Vulnerability Database. CVE statistics. http://web. nvd.nist.gov/view/vuln/statistics, February 2011.

[33]V. H. Nguyen, T. K. Dang, N. T. Son, and J. Kung. Query as- surance verification for dynamic outsourced XML databases. In *Proceedings of the 2nd Conference on Availability, Reliability and Security*, Vienna, Austria, April 2007.
[34] Oracle Corporation. Oracle advanced security. http: //www.oracle.com/technetwork/database/options/ advanced-security/.
[35]P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 18th Annual Inter- national Conference on the Theory and Applications of Cryp- tographic Techniques (EUROCRYPT)*, Prague, Czech Republic, May 1999.
[36]B. Parno, J. M. McCune, D. Wendlandt, D. G. Andersen, and A. Perrig. CLAMP: Practical prevention of large-scale data leaks. In *Proceedings of the 30th IEEE Symposium on Security and Privacy*, Oakland, CA, May 2009.
[37]R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakr- ishnan. CryptDB web site. http://css.csail.mit.edu/ cryptdb/.
[38]R. A. Popa, J. R. Lorch, D. Molnar, H. J. Wang, and L. Zhuang. Enabling security in cloud storage SLAs with CloudProof. In *Proceedings of 2011 USENIX Annual Technical Conference*, Portland, OR, 2011. [39]R. A. Popa, N. Zeldovich, and H. Balakrishnan. CryptDB: A practical encrypted relational DBMS. Technical Report MIT- CSAIL-TR-2011-005, MIT Computer Science and Artificial In- telligence Laboratory, Cambridge, MA, January 2011.
[40]Privacy Rights Clearinghouse. Chronology of data breaches. http://www.privacyrights.org/data-breach.
[41]S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy. Extending query rewriting techniques for fine-grained access control. In *Proceedings of the 2004 ACM SIGMOD International Confer- ence on Management of Data*, Paris, France, June 2004.
[42]H. Shacham, N. Modadugu, and D. Boneh. Sirius: Securing remote untrusted storage. In *Proceedings of the 10th Network and Distributed System Security Symposium*, 2003.
[43]E. Shi, J. Bethencourt, H. Chan, D. Song, and A. Perrig. Multi- dimensional range query over encrypted data. In *Proceedings of the 28th IEEE Symposium on Security and Privacy*, Oakland, CA, May 2007.
[44]V. Shoup. NTL: A library for doing number theory. http:// www.shoup.net/ntl/, August 2009.
[45]R. Sion. Query execution assurance for outsourced databases. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 601–612, Trondheim, Norway, August– September 2005.
[46]D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the 21st IEEE Symposium on Security and Privacy*, Oakland, CA, May 2000.
[47]M. Taylor. MySQL proxy. https://launchpad.net/ mysql-proxy.
[48]B. Thompson, S. Haber, W. G. Horne, T. S, and D. Yao. Privacy- preserving computation and verification of aggregate queries on outsourced databases. Technical Report HPL-2009-119, HP Labs, 2009.
[49]E. P. Wobber, M. Abadi, M. Burrows, and B. Lampson. Au- thentication in the Taos operating system. *ACM Transactions on Computer Systems*, 12(1):3–32, 1994.
[50]L. Xiong, S. Chitti, and L. Liu. Preserving data privacy for out- sourcing data aggregation services. Technical Report TR-2007- 013, Emory University, Department of Mathematics and Com- puter Science, 2007.
[51]Z. Yang, S. Zhong, and R. N. Wright. Privacy-preserving queries on encrypted data. In *European Symposium on Research in Computer Security*, 2006.
[52]A. Yip, X. Wang, N. Zeldovich, and M. F. Kaashoek. Improving application security with data flow assertions. In *Proceedings of the 22nd ACM Symposium on Operating Systems Principles*, pages 291–304, Big Sky, MT, October 2009.