15EC401M – MULTI DISCIPLINARY DESIGN

On

**Cloud security system using an FPGA interfaced IaaS** *Submitted for partial fulfilment for the award of the degree* **BACHELOR OF TECHNOLOGY**

In

## ELECTRONICS AND COMMUNICATION ENGINEERING

*Submitted By*

|  |  |
| --- | --- |
| **K. PRAVEEN** | **RA1711004040019** |
| **B.L. AMRUTH LAKSHMAN** | **RA1711004040038** |
| **R. RAGHUL** | **RA1711004040051** |



Under the guidance of

## MR. DINESH BABU A

(Assistant Professor, Department of Electronics and Communication Engineering)

## DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING,

**FACULTY OF ENGINEERING AND TECHNOLOGY,**

## SRM INSTITUTE OF SCIENCE & TECHNOLOGY, VADAPALANI CAMPUS,

**CHENNAI-600 026,**

**DEC 2020.**

# BONAFIDE CERTIFICATE

Certified that this project report titled “**CLOUD SECURITY SYSTEM USING AN FPGA INTERFACED IaaS**” is the bonafide work of **PRAVEEN K (RA1711004040019), AMRUTH LAKSHMAN B L (RA1711004040038)** and **RAGHUL R (RA1711004040051)** who carried out

the project work under my supervision as a batch. Certified further, that to the best of my knowledge the work reported herein does not form any other project on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

|  |  |
| --- | --- |
| **Project guide** | **Head of the Department** |
| Mr. Dinesh Babu A | Dr. C. Gomathy, |
| Asst. Professor, | Professor and Head, |
| Department of ECE, | Department of ECE, |
| SRM IST. | SRM IST. |

*Submitted for University Examination held in December 2020 at SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, Vadapalani, Chennai-600 026.*

Date: Internal Examiner External Examiner

# ACKNOWLEDGEMENT

This work would not have been completed without motivation and help from our **SRM IST MANAGEMENT**. We place our deep sense of gratitude to our beloved chancellor, **Dr. T.R.PACHAMUTHU, SRM IST** for providing us with the requisite infrastructure throughout the course, We would like to thank our respected **Dr. DURAIVELU**, Dean (E&T), SRM IST, Vadapalani rendering all facilities to do our project. We also convey our immense gratitude **to Dr. C.GOMATHY**, Head of the Department, Department of ECE for her constant support and encouragement rendered to us for completion of our project.

We would like to have a special mention of our class in-charge and guide **Mr. A.DINESH BABU**, Assistant professor, Department of ECE for his efficient and excellent guidance, inspiring discussions insightful and timely encouragements for the successful completion of our project.

We also extend our heartfelt thanks to all the teaching and non-teaching staff members of ECE Department who have rendered their valuable help in making for successful completion of the project. We could not have finished this work without support from our family and friends. They are dearest in our heart and a great source of motivation. Our hearty appreciation and thanks to our friends whose support and encouragement made all this possible.

|  |  |  |
| --- | --- | --- |
|  | **TABLE OF CONTENTS** |  |
| **ACKNOWLEDGEMENT** | | **iii** |
| **LIST OF FIGURES** | | **viii** |
| **LIST OF TABLES.** | | **x** |
| **LIST OF ABBREVIATION** | | **xi** |
| **ABSTRACT** |  | **xii** |
|  | **INDEX** |  |
| **CHAPTER NO.** | **CHAPTER NAME** | **PAGE NO.** |
| **1** | **INTRODUCTION** |  |
|  | 1.1 Objective | 2 |
|  | 1.2 Formulation of the problem | 2 |
|  | 1.3 Chapter Organisation | 3 |
| **2** | **LITERATURE REVIEW** |  |
|  | 2.1 Literature Survey | 4 |
| 2.1.1 Zhuangdi Zhu et al.FPGA Resource | 4 |
| Pooling in Cloud |  |
| Computing |  |
| 2.1.2 Mark A. Will et al., Secure | 4 |
| FPGA as a Service Towards |  |
| Secure Data Processing by |  |
| Physicalizing the Cloud |  |
| 2.1.3 Ashish Singh et al., Cloud | 5 |
| security issues and |  |
| challenges: A survey |  |

|  |  |  |
| --- | --- | --- |
|  | 2.1.4 Radosław Cieszewski et al.,  Review of parallel computing methods and tools for FPGA technology | 5 |
|  | 2.1.5 Katherine Compton et., An Introduction to Reconfigurable Computing | 5 |
| **3** | **DOMAINS** |  |
|  | * 1. Cloud      1. Software-as-a-Service      2. Infrastructure-as-a-service      3. Platform-as-a-Service      4. Function-as-a-service      5. Amazon Web Services      6. Amazon S3      7. Identity Access Management | 7 |
| 10 |
| 10 |
| 11 |
| 12  xx  xxx  xxx |
|  | * 1. Cryptography      1. Techniques Used      2. Components of crypto system      3. Encryption         1. Symmetric Encryption         2. Asymmetric Encryption            1. The two main   types of asymmetric encryption.   * + - 1. Hybrid Encryption     1. Key differences between cryptography and encryption     2. Decryption        1. Decryption Usage | 14 |
| 14 |
| 15 |
| 16 |
| 16 |
| 19 |
| 20 |
| 22 |
| 22 |
| 23 |
| 24 |

|  |  |  |
| --- | --- | --- |
|  | 3.3 Re Configurable Computing | 24 |
|  | 3.4 Parallel Computing | 27 |
|  | * 1. ReactJS      1. What is React      2. Why React      3. Features of React      4. Components      5. States      6. Props |  |
| **4** | **PROJECT APPARATUS** |  |
|  | * 1. FPGA      1. FPGA architecture      2. Types of FPGA      3. ASIC vs FPGA      4. FPGA Applications      5. FPGA for cloud infrastructure      6. FPGA Challenges      7. Cloud FPGA services      8. FPGA PaaS | 28 |
| 28 |
| 29 |
| 31 |
| 31 |
| 33 |
| 34 |
| 38 |
| 39 |
|  | * 1. Raspberry Pi      1. Working of R-pi | 42 |
| 43 |
|  | 4.3 SD Card | 44 |
| **5** | **MDD VIEWPOINTS** |  |
|  | 5.1 Major types of thinking required in the system | 45 |
|  | 5.2 System Modelling | 46 |
|  | 5.3 Decomposing the system. Identifying the major subsystems. | 47 |
|  | * 1. Mathematical Modelling and governing equations of the system.      1. Alphabets and Texts      2. Ciphers | 48 |
| 49 |
| 49 |

|  |  |  |
| --- | --- | --- |
|  | 5.4.3 AES | 50 |
|  | 5.5 Design Variables | 51 |
|  | 5.6 Constraints | 52 |
|  | * 1. Conceptual Designs      1. Merging the Phases      2. Manage      3. Explore      4. Create      5. Evaluate | 55  55  57  58  59  60 |
|  | * 1. Costing and financial model      1. Types of project Cost | 62  62 |
| **6** | **CONCLUSIONS** |  |
|  | 6.1 Limitations | 64 |
|  | 6.2 Future Scope | 64 |
|  | **REFERENCES** | 65 |

|  |  |  |
| --- | --- | --- |
|  | **LIST OF FIGURES** |  |
| **FIG NO.** | **NAME OF FIGURES** | **PAGE NO.** |
| 3.1 | AWS S3 | 15 |
| 3.2 | Cryptography Building Blocks | 15 |
| 3.3 | A programming bit for SRAM- based FPGAs | 26 |
| 3.4 | Programmable routing connection | 26 |
| 3.5 | D flip-flop with optional bypass | 27 |
| 3.6 | A 3-input LUT | 27 |
| 3.7 | MVC Architecture | 15 |
| 3.8 | Instagram Components |  |
| 3.9 | React Features |  |
| 3.10 | One-way Data binding |  |
| 3.11 | React Components |  |
| 4.1 | Symmetrical array architecture | 30 |
| 4.2 | Row based architecture | 31 |
| 4.3 | Hierarchical Architecture | 31 |
| 4.4 | Interactions between application designers and hardware designers for FPGA design and implementation | 36 |
| 4.5 | FPGA design flow with parallel parameters exploration. | 37 |

|  |  |  |
| --- | --- | --- |
| 4.6 | FPGA Platform as a Service  model. | 41 |
| 4.7 | Raspberry Pi 4 | 43 |
| 4.8 | SanDisk Extreme 64GB | 45 |
| 5.1 | Architectural modelling of the system | 47 |
| 5.2 | Architecture Modelling of Subsystems | 48 |
| 5.3 | The fundamental questions of design | 56 |
| 5.4 | Manage cycles | 57 |
| 5.5 | Manage activities | 58 |
| 5.6 | Explore activities | 60 |
| 5.7 | Create activities | 61 |
| 5.8 | Evaluate activities | 62 |

|  |  |  |
| --- | --- | --- |
|  | **LIST OF TABLES** |  |
| **TABLE NO.** | **TABLE NAME** | **PAGE NO.** |
| 3.1 | Symmetric vs Asymmetric Encryption | 22 |
| 4.1 | Raspberry pi pin configuration | 44 |
| 5.1 | Cost model for the prototype | 64 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **ACRONYM** | **ABBREVIATIONS** |
| SaaS | Software-as-a-service |
| IaaS | Infrastructure-as-a-service |
| PaaS | Platform-as-a-Service |
| FaaS | Function-as-Service |
| AES | Advanced Encryption Standard |
| DES | Data Encryption Standard |
| RSA | Rivest-Shamir-Adleman |
| ECC | Elliptic Curve Cryptography |
| TLS | Transport Layer Security |
| FPGA | Field Programmable Gate Array |
| PKI | Public Key Infrastructure |
| PAL | Programmable Array Logic |
| MPGA | Mask Programmable Gate Arrays |
| LUT | Look Up Table |
| CLB | Configurable Logic Blocks |
| ASIC | Application Specific Integrated Circuits |
| CPU | Central Processing Unit |
| RAM | Random Access Memory |
| GPIO | General Purpose Input/Output |
| IoT | Internet of Things |

# ABSTRACT

In the fast-growing technology industry, the cloud domain and cloud computing has contributed and is playing a major role. The security of the cloud is one of the most important factors that needs to be focused on. In the corporate world there are more than a billion misconfigurations happening each year. In this project we propose an approach to improve the cloud computing security by introducing an extra layer of hardware which is the FPGA. We also work on advanced encryption standards like the AES, DES and RSA for an improvised security system. With this enhanced security in the cloud computing, data breaches and misconfigurations will be minimised to a greater extent thereby making the cloud platforms more reliable and trustworthy. The data will get encrypted in the cloud but with our model the data is encrypted and then sent to the cloud which makes the data double encrypted thereby making it more secure. For this we are going to use hardware’s such as FPGA, raspberry pi and SD card and software techniques such as reconfigurable computing and encryption / decryption techniques like AES, DES and RSA.

# CHAPTER 1 INTRODUCTION

Cloud computing is another name for an old concept. The computing service is implemented from a remote location. Cloud computing is Internet-based processing, where shared resources and data are given to PCs and various devices on request. Cloud hosting enables companies to maintain the same systems and business measures without modifying the backend technicalities. Cloud infrastructure, which is easily manageable via the Internet, will be accessed by enterprises to easily and quickly access user data anywhere, at any time. An Internet cloud infrastructure boosts business productivity and efficiency by ensuring the consistency of user applications. This takes into account direct collaboration as well as client division in various areas. The Bane, on the other hand, comes with the assistance. The host organisation obtains user data on a regular basis, with or without consent. The specialist organisation can access the information in the cloud at any time. They may inadvertently or intentionally alter or delete data.Outsiders are used in cloud-based administrations for capacity and security. Would one be able to accept that a cloud-based organisation will guarantee and get one's information if one uses their administrations at a very low or no cost? They may disclose client information to others. Security is a real threat to the cloud.While providing cloud administrations, it should be ensured that the customer is not purchasing distributed computing administrations for nefarious purposes. Meeting riding occurs when an aggressor takes a client's treat and uses the machine in the client's name. An attacker may also use CSRF attacks to trick the client into sending confirmed requests to discretion. In 2009, a financial Trojan wrongfully utilized the supported Amazon administration as an order and control channel that gave programming refreshes and malevolent guidance to PCs that were tainted by the malware. Utilizing information from a 2019 Ponemon Institute report that said the commonplace expense per lost record around the world is $150, DivvyCloud specialists assessed that cloud misconfiguration penetrates cost organizations upwards of $5 trillion over those two years. Specialists from the cloud security organization DivvyCloud found that breaks brought about by cloud misconfigurations cost organizations worldwide an expected $5 trillion of every 2018 and 2019. More than 33 billion records have been uncovered throughout the most recent two years as a great many organizations move to cloud conditions without the suitable security frameworks set up. [18] Most distributed computing security hazards are identified with cloud information security. In virtualized conditions, the actual workers run numerous virtual machines on top of hypervisors. An aggressor can misuse a hypervisor distantly by utilizing a weakness present inside the hypervisor itself – such weaknesses are very uncommon, however, they are doing exist. Moreover, a virtual machine can shake the virtualized sandbox climate and access the hypervisor and considerably every one of the virtual machines running consequently. Regardless of whether a shortage of perceivability to information, failure to direct information, or robbery of information inside the cloud, most issues go to the data clients put inside the cloud.

## OBJECTIVE

Safely handling information in the cloud by utilizing equipment like the FPGA as an extra layer of safety and giving more noteworthy execution over the product arrangements. Utilizing this equipment layer to give different encoding of a similar information record.

To utilize different encryption methods like progressed encryption standard (AES), information encryption standard (DES), and Rivest–Shamir–Adleman (RSA) to propel the encryption/unscrambling guidelines of the model.

## FORMULATION OF THE PROBLEM

Planning a drag detailing is that the first and most essential advance of a pursuit interaction. The difficult detailing resembles a distinguishing proof of an objective prior to undertaking an excursion. The exploration issue is an establishment of an inquiry study; kind of a structure and its establishment.

The main distributed computing information security issues are numerous genuine threats such as infection attack and hacking of the customer's site. Before receiving distributed computing innovation for their business, business visionaries must consider these issues. Because the client is entrusting the organization's most sensitive details to an outsider, it is critical to ensure the sensitivity. The organization's high authority and hierarchical culture has also become a major impediment to the proper execution of distributed computing. Cryptography calculations frequently necessitate the use of random number generators, which use erratic data sources to generate real irregular numbers, which are required to generate a large entropy pool.If the random number generators produce only a small amount of entropy, the numbers can be beast constrained. The primary source of randomization in customer PCs is client mouse development and key presses; however, workers are generally operating without client communication, implying a significantly lower number of randomization sources. As a result, the virtual machines should rely on the sources that are available to them, which could result in effectively guessable numbers that do not provide a lot of entropy in cryptographic calculations. When transferring data from customers to the cloud, a scrambled secure channel, such as SSL/TLS, should be used.This forestalls various assaults like MITM assaults, where the information could be taken by an aggressor capturing the correspondence. Different cloud administrations on the web are uncovered by application programming interfaces. Since the APIs are open from any place on the web, vindictive aggressors can utilize them to bargain the secrecy and trustworthiness of the undertaking clients. An aggressor acquiring a token utilized by a client to get to the assistance through help API can utilize an identical token to control the client's information. Subsequently, it's basic that cloud administrations give a protected API, delivering such assaults useless. The cloud administration SaaS/PasS/IaaS suppliers utilize an adaptable framework to help numerous inhabitants who share the fundamental foundation. Straightforwardly on the equipment layer, there are hypervisors running various virtual machines, themselves running different applications. The information put away inside the cloud may be lost because of the plate drive disappointment. A CSP could incidentally erase the information, an aggressor may alter the data, and so forth Thusly, the easiest gratitude to secure against information misfortune is by having the right information reinforcement, which takes care of the data misfortune issues. Information misfortune can have calamitous results to the business, which can end in a business chapter 11, which is the reason keeping the data upheld up is normally the least complex choice. Top authority never needs to store the significant information of the corporate somewhere else where they're not able to direct and get to the data. They have confusions to them that distributed computing puts the association at threat by leaking out significant subtleties. Their attitude is such the association on hazard unwilling balance, which makes it more hesitant to move to a cloud arrangement. The deficiency of assets and skill is one of the cloud relocation challenges this year. According to the report by RightScale, practically 75% of the respondents stamped it as a test while 23% said that it was a genuine test. Consistency is also one of the challenges that distributed computing will face in 2020. This is frequently a stumbling block for anyone who uses distributed storage or reinforcement services. When an organisation moves information from its internal storage to the cloud, it must comply with the business's laws and guidelines.

## CHAPTER ORGANIZATION

* + 1. Chapter Two discusses the current knowledge including substantive findings as well as theoretical and methodological contributions from the literature survey.
    2. Chapter Three discusses various domain knowledge and understanding of major topics under each domain.
    3. Chapter Four discusses the apparatus used in the proposed system.
    4. Chapter Five discusses the approach seeks to integrate various skills and methodology from multiple disciplines into a collaborative effort.
    5. Chapter Six concludes the proposal and discusses the future scope of the project.

# CHAPTER 2 LITERATURE REVIEW

## LITERATURE SURVEY

### Zhuangdi Zhu et al., FPGA Resource Pooling in Cloud Computing

Cloud providers have started to deploy various FPGA accelerators in their datacentres because the performance of many applications can be significantly improved by implementing their core routines in FPGAs. In conventional datacentres with FPGA accelerated servers, if a tenant wants to use FPGA accelerators, it requests for a VM instance residing in a server equipped with an FPGA accelerator. This paradigm to integrate FPGA into Cloud leads to poor resource sharing of the precious FPGA resources. In this paper, they have proposed FPGAPooling, an FPAG-enabled Cloud system where all FPGA accelerators are managed as a single resource pool and shared among all VMs. For a VM, instead of requesting the Cloud to run the VM on an FPGA accelerated server, at runtime, when a VM needs to use FPGA acceleration, it requests an FPGA accelerator from the pool. They have designed a centralized scheduler to handle acceleration requests at runtime; They have implemented a system prototype on IBM's OpenPower Cloud system. They have also designed and implemented a group of scheduling algorithms for the FPGAPooling system. With extensive evaluations, they have found that their algorithms can improve the average and tail job completion time by up to 7 and 4 times, respectively.

### Mark A. Will et al., Secure FPGA as a Service — Towards Secure Data Processing by Physicalizing the Cloud

Securely processing data in the cloud is still a difficult problem, even with homomorphic encryption and other privacy preserving schemes. Hardware solutions provide additional layers of security and greater performance over their software alternatives. However, by definition the cloud should be flexible and adaptive, often viewed as abstracting services from products. By creating services reliant on custom hardware, the core essence of the cloud is lost. FPGAs bridge this gap between software and hardware with programmable logic, allowing the cloud to remain abstract. FPGA as a Service (FaaS) has been proposed for a greener cloud, but not for secure data processing. This paper explores the possibility of Secure FaaS in the cloud for privacy preserving data

processing, describes the technologies required, identifies use cases, and highlights potential challenges.

### Ashish Singh et al., Cloud security issues and challenges: A survey

Cloud computing provides on demand services over the Internet with the help of a large amount of virtual storage. The main features of cloud computing is that the user does not have any setup of expensive computing infrastructure and the cost of its services is less. In recent years, cloud computing integrates with the industry and many other areas, which has been encouraging the researcher to research new related technologies. Due to the availability of its services & scalability for computing processes individual users and organizations transfer their application, data and services to the cloud storage server. Regardless of its advantages, the transformation of local computing to remote computing has brought many security issues and challenges for both consumer and provider. Many cloud services are provided by the trusted third party which arises new security threats. The cloud provider provides its services through the Internet and uses many web technologies that arise new security issues. This paper discussed the basic features of cloud computing, security issues, threats and their solutions. Additionally, the paper describes several key topics related to the cloud, namely cloud architecture framework, service and deployment model, cloud technologies, cloud security concepts, threats, and attacks. The paper also discusses a lot of open research issues related to cloud security.

### Radosław Cieszewski et al., Review of parallel computing methods and tools for FPGA technology

Parallel computing is emerging as an important area of research in computer architectures and software systems. Many algorithms can be greatly accelerated using parallel computing techniques. Specialized parallel computer architectures are used for accelerating specific tasks. High-Energy Physics Experiments measuring systems often use FPGAs for ne-grained computation. FPGA combines many benefits of both software and ASIC implementations. Like software, the mapped circuit is exible, and can be reconfigured over the lifetime of the system. FPGAs therefore have the potential to achieve far greater performance than software as a result of bypassing the fetch-decode-execute operations of traditional processors, and possibly exploiting a greater level of parallelism. Creating implemented in FPGAs is not trivial. This paper presents

existing methods and tools for ne-grained computation implemented in FPGA using Behavioral Description and High Level Programming Languages.

### Katherine Compton et., An Introduction to Reconfigurable Computing

Due to its potential to greatly accelerate a wide variety of applications, reconfigurable computing has become a subject of a great deal of research. Its key feature is the ability to perform computations in hardware to increase performance, while retaining much of the flexibility of a software solution. In this introduction to reconfigurable computing, they give an overview of the hardware architectures of reconfigurable computing machines, and the software that targets these machines, such as compilation tools. Finally, they consider the issues involved in run-time reconfigurable systems, which re-use the configurable hardware during program execution.

# CHAPTER 3 DOMAINS

## CLOUD

The cloud refers to any type of software or service that isn't located on any personal computer or devices but instead runs on the internet. The files, images and videos that gets saved on cloud services are stored on the servers of third parties, companies such as Amazon, Google, and Microsoft. The cloud enables users to access the same files and applications from almost any device, because the computing and storage takes place on servers in a data centre, instead of locally on the user device. This is why a user can log into their Instagram account on a new phone after their old phone breaks and still find their old account in place, with all their photos, videos, and conversation history. It works the same way with cloud email providers like Gmail or Microsoft Office 365, and with cloud storage providers like Dropbox or Google Drive. “The cloud" started off as a tech industry slang term. In the early days of the Internet, technical diagrams often represented the servers and networking infrastructure that make up the Internet as a cloud. As more computing processes moved to this servers-and-infrastructure part of the Internet, people began to talk about moving to "the cloud" as a shorthand way of expressing where the computing processes were taking place. Today, "the cloud" is a widely accepted term for this style of computing. The Internet has always been made up of servers, clients, and the infrastructure that connects them. Clients make requests of servers, and servers send responses. Cloud computing differs from this model in that cloud servers aren't just responding to requests – they're running programs and storing data on the client's behalf.

For businesses, switching to cloud computing removes some IT costs and overhead: for instance, they no longer need to update and maintain their own servers, as the cloud vendor they are using will do that. This especially makes an impact for small businesses that may not have been able to afford their own internal infrastructure but can outsource their infrastructure needs affordably via the cloud. The cloud can also make it easier for companies to operate internationally, because employees and customers can access the same files and applications from any location.

Cloud computing is possible because of a technology called virtualization. Virtualization allows for the creation of a simulated, digital-only "virtual" computer that behaves as if it were a

physical computer with its own hardware. The technical term for such a computer is virtual machine. When properly implemented, virtual machines on the same host machine are sandboxed from one another, so they don't interact with each other at all, and the files and applications from one virtual machine aren't visible to the other virtual machines even though they're on the same physical machine.[1]

Virtual machines also make more efficient use of the hardware hosting them. By running many virtual machines at once, one server becomes many servers, and a data center becomes a whole host of data centres, able to serve many organizations. Thus, cloud providers can offer the use of their servers to far more customers at once than they would be able to otherwise, and they can do so at a low cost. Even if individual servers go down, cloud servers in general should be always online and always available. Cloud vendors generally back up their services on multiple machines and across multiple regions.

Users access cloud services either through a browser or through an app, connecting to the cloud over the Internet – that is, through many interconnected networks – regardless of what device they're using. The most common cloud deployments are:

**Private cloud**: A private cloud is a server, data centre, or distributed network wholly dedicated to one organization.

**Public cloud**: A public cloud is a service run by an external vendor that may include servers in one or multiple data centres. Unlike a private cloud, public clouds are shared by multiple organizations. Using virtual machines, individual servers may be shared by different companies, a situation that is called "multi tenancy" because multiple tenants are renting server space within the same server. **Hybrid cloud**: Hybrid cloud deployments combine public and private clouds, and may even include on-premises legacy servers. An organization may use their private cloud for some services and their public cloud for others, or they may use the public cloud as backup for their private cloud.

**Multi Cloud**: Multiload is a type of cloud deployment that involves using multiple public clouds. In other words, an organization with a multiload deployment rents virtual servers and services from several external vendors – to continue the analogy used above, this is like leasing several adjacent plots of land from different landlords. Multi Cloud deployments can also be hybrid cloud, and vice versa. Cost reductions are claimed by cloud providers. A public-cloud delivery model converts capital expenditures (e.g., buying servers) to operational expenditure. This purportedly lowers barriers to entry, as infrastructure is typically provided by a third party and need not be purchased for one-time or infrequent intensive computing tasks. Pricing on a utility computing basis is "fine-

grained", with usage-based billing options. As well, less in-house IT skills are required for implementation of projects that use cloud computing. The e-FISCAL project's state-of-the-art repository contains several articles looking into cost aspects in more detail, most of them concluding that costs savings depend on the type of activities supported and the type of infrastructure available in-house.

Device and location independence enable users to access systems using a web browser regardless of their location or what device they use (e.g., PC, mobile phone). As infrastructure is off-site (typically provided by a third-party) and accessed via the Internet, users can connect to it from anywhere. Maintenance of cloud computing applications is easier, because they do not need to be installed on each user's computer and can be accessed from different places (e.g., different work locations, while travelling, etc.).

Multitenancy enables sharing of resources and costs across a large pool of users thus allowing for: centralization of infrastructure in locations with lower costs (such as real estate, electricity, etc.) peak-load capacity increases utilisation and efficiency improvements for systems that are often only 10–20% utilised. Productivity may be increased when multiple users can work on the same data simultaneously, rather than waiting for it to be saved and emailed. Time may be saved as information does not need to be re-entered when fields are matched, nor do users need to install application software upgrades to their computer.

Availability improves with the use of multiple redundant sites, which makes well-designed cloud computing suitable for business continuity and disaster recovery. Scalability and elasticity via dynamic provisioning of resources on a fine-grained, self-service basis in near real-time without users having to engineer for peak loads. This gives the ability to scale up when the usage need increases or down if resources are not being used. Emerging approaches for managing elasticity include the use of machine learning techniques to propose efficient elasticity models.

Security can improve due to centralization of data, increased security-focused resources, etc., but concerns can persist about loss of control over certain sensitive data, and the lack of security for stored kernels. Security is often as good as or better than other traditional systems, in part because service providers are able to devote resources to solving security issues that many customers cannot afford to tackle or which they lack the technical skills to address. However, the complexity of security is greatly increased when data is distributed over a wider area or over a greater number of devices, as well as in multi-tenant systems shared by unrelated users. In addition, user access to security audit logs may be difficult or impossible. Private cloud installations are in part motivated by users' desire to retain control over the infrastructure and avoid losing control of information security.[2]

#### Service Models as Cloud

* + 1. SaaS
    2. Paas
    3. Iaas
    4. Faas

#### Software-as-a-Service (SaaS)

Instead of users installing an application on their device, SaaS applications are hosted on cloud servers, and users access them over the Internet. SaaS is like renting a house: the landlord maintains the house, but the tenant mostly gets to use it as if they owned it. Examples of SaaS applications include Salesforce, MailChimp, and Slack. In the software on demand SaaS model, the provider gives customers network-based access to a single copy of an application that the provider created specifically for SaaS distribution. The application’s source code is the same for all customers and when new features or functionalities are rolled out, they are rolled out to all customers. Depending upon the service level agreement (SLA), the customer’s data for each model may be stored locally, in the cloud or both locally and in the cloud.

Organizations can integrate SaaS applications with other software using application programming interfaces (APIs). For example, a business can write its own software tools and use the SaaS provider's APIs to integrate those tools with the SaaS offering.[3]

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

Typically, IaaS involves the use of a cloud orchestration technology like OpenStack, Apache Cloudstack or OpenNebula. This manages the creation of a virtual machine and decides on which hypervisor (i.e. physical host) to start it, enables VM migration features between hosts, allocates storage volumes and attaches them to VMs, usage information for billing and lots more.IaaS cloud computing platform cannot replace the traditional hosting method, but it provides more than that, and each resource which are used are predictable as per the usage.

IaaS cloud computing platform may not eliminate the need for an in-house IT department. It will be needed to monitor or control the IaaS setup. IT salary expenditure might not reduce significantly, but other IT expenses can be reduced.

Breakdowns at the IaaS cloud computing platform vendor's can bring your business to the halt stage. Assess the IaaS cloud computing platform vendor's stability and finances. Make sure that SLAs (i.e., Service Level Agreement) provide backups for data, hardware, network, and application failures. Image portability and third-party support is a plus point.

An alternative to hypervisors are Linux containers, which run in isolated partitions of a single Linux kernel running directly on the physical hardware. Linux groups and namespaces are the underlying Linux kernel technologies used to isolate, secure and manage the containers. Containerisation offers higher performance than virtualization, because there is no hypervisor overhead. Also, container capacity auto-scales dynamically with computing load, which eliminates the problem of over-provisioning and enables usage-based billing.

IaaS clouds often offer additional resources such as a virtual-machine disk-image library, raw block storage, file or object storage, firewalls, load balancers, IP addresses, virtual local area networks (VLANs), and software bundles. The IaaS cloud computing platform cannot replace the traditional hosting method, but it provides more than that, and each resource which is used is predictable as per the usage. IaaS cloud computing platform may not eliminate the need for an in- house IT department. It will be needed to monitor or control the IaaS setup. IT salary expenditure might not reduce significantly, but other IT expenses can be reduced. Breakdowns at the IaaS cloud computing platform vendor's can brings the business to the halt stage. Assess the IaaS cloud computing platform vendor's stability and finances. Make sure that SLAs (i.e., Service Level Agreement) provide backups for data, hardware, network, and application failures. Image portability and third-party support is a plus point.

The IaaS cloud computing platform vendor can get access to the sensitive data. So, engage with credible companies or organizations. Study their security policies and precautions. It's much easier to expand a business with IaaS as the foundation. Instead of purchasing, installing, and maintaining a new server every time the business needs to scale up, they can just add a new server on demand through the IaaS provider. This on-demand scalability is a major benefit of cloud computing across all cloud service models. With IaaS, a company has essentially outsourced server purchasing, maintenance, and updating to the IaaS provider. This is typically cheaper and requires less time and labour from internal teams than they would need to host their own infrastructure. Companies using IaaS can deploy and update applications much faster, since cloud providers can offer however much infrastructure they need as they need it.[4]

#### Platform-as-a-Service (PaaS)

In this model, companies don't pay for hosted applications; instead they pay for the things they need to build their own applications. PaaS vendors offer everything necessary for building an application, including development tools, infrastructure, and operating systems, over the Internet. PaaS can be compared to renting all the tools and equipment necessary for building a house, instead of renting the house itself. PaaS examples include Heroku and Microsoft Azure. Because PaaS architectures keep the underlying infrastructure out of sight of developers and other users, the model is similar to the concepts of serverless computing and function-as-a-service (FaaS), in which a cloud service provider provisions and runs the server and manages the allocation of resources.

One of the biggest advantages of PaaS is that enterprises can gain an environment in which to create and deploy new applications without the need to spend time and money building and

maintaining an infrastructure that includes servers and databases. This can lead to faster development and delivery of applications, a huge plus for businesses looking to gain a competitive edge or that need to get products to market quickly.

PaaS also lets them test the use of new languages, operating systems, databases, and other development technologies quickly, because they do not have to stand up the supporting infrastructure for them. PaaS also makes it easier and faster to upgrade their tools and the use of PaaS forces enterprise software developers to use cloud techniques in their applications, helping then adopt modern principles and take better advantage of cloud infrastructure (IaaS) platforms. Because organizations using PaaS can manage their applications and data, loss of control is not a major issue as it often is when using cloud infrastructure or applications. FaaS is a type of serverless offering that allows companies to develop and run discrete, event-driven functions without the complexity of building and maintaining the infrastructure typically needed for developing and launching an application.

PaaS and serverless computing services typically charge only for compute, storage, and network resources consumed. FaaS takes that approach to the extreme, charging only when functions are executed, making FaaS a natural choice for intermittent tasks.

Formerly, SaaS, PaaS, and IaaS were the three main models of cloud computing, and essentially all cloud services fit into one of these categories. However, in recent years a fourth model has emerged.[5]

* + 1. **Function-as-a-Service (FaaS)**

FaaS, also known as serverless computing, breaks cloud applications down into even smaller components that only run when they're needed. Imagine if it were possible to rent a house one little bit at a time: for instance, the tenant only pays for the dining room at dinner time, the bedroom while they're sleeping, the living room while they're watching TV, and when they aren't using those rooms, they don't have to pay rent on them. To fully appreciate the FaaS model and how it can benefit software developers, we need to clarify two important bits of terminology. Serverless architecture does not literally mean that the application runs without a server. It should be obvious that any application deployment requires some kind of hardware host. The defining feature of serverless architecture or serverless computing is that a cloud service provider takes

responsibility for managing the application servers and dynamically allocating storage space based on the needs of users. In this sense, serverless architecture is only serverless for the developer - cloud service providers are still very much involved in the management of servers. A function is a task or operation within an application that can be written as a discrete piece of code and executed independently. Functions as extensions of the microservice architecture: Early applications were written using monolithic architecture. The application was structured as a single executable that had to be triggered all at once. As best practices evolved, software developers increasingly adopted a microservice architecture for application design. Applications were re-imagined as a modular collection of microservices that would be easier to test and maintain and independently deployable. If a microservice can only perform one action in response to an event, it could be considered a function. More complex microservices that can perform several actions may be composed of several functions.

FaaS or serverless applications still run on servers, as do all these models of cloud computing. But they're called "serverless" because they don't run on dedicated machines, and because the companies building the applications don't have to manage any servers. FaaS is a cloud service execution model, so an organization that wishes to take advantage of FaaS should start by forming a relationship with a cloud service provider that advertises FaaS capabilities. In the FaaS delivery model, developers have no responsibility for maintaining application servers. Instead, they are hosted externally by the service provider and allocated dynamically based on the needs of the customer. FaaS service providers enable software developers to deploy pieces of code known as functions that can be executed on-demand.

When the function is called, the service provider spins up a server, executes the function, then shuts down. Unlike other models where software developers run the application on a dedicated server, serverless architecture is only active when the function is actively being used. Once the function has been executed, it can be shut down again, allowing the same computing resources to be allocated elsewhere. With functions-as-a-service, software developers have access to a platform that executes application logic on demand and where all of the application resources are secured and coordinated by the service provider. The FaaS model works best for simple, repetitive functions such as scheduled routine tasks or jobs, processing web requests or processing queue messages.[6]

**3.1.5 Amazon Web Services**

Amazon web service is an online platform that provides scalable and cost-effective cloud computing solutions.

AWS is a broadly adopted cloud platform that offers several on-demand operations like compute power, database storage, content delivery, etc., to help corporates scale and grow.

AWS is a comprehensive, easy to use computing platform offered Amazon. The platform is developed with a combination of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offerings.

Here, are Cloud Compute Services offered by Amazon:

* EC2(Elastic Compute Cloud) - EC2 is a virtual machine in the cloud on which you have OS level control. You can run this cloud server whenever you want.
* LightSail?-This cloud computing tool automatically deploys and manages the computer, storage, and networking capabilities required to run your applications.
* Elastic Beanstalk?—? The tool offers automated deployment and provisioning of resources like a highly scalable production website.
* EKS (Elastic Container Service for Kubernetes)?—?The tool allows you to Kubernetes on Amazon cloud environment without installation.
* AWS Lambda?—?This AWS service allows you to run functions in the cloud. The tool is a big cost saver for you as you to pay only when your functions execute.
* IAM (Identity and Access Management)?—? IAM is a secure cloud security service which helps you to manage users, assign policies, form groups to manage multiple users.
* Inspector?—?It is an agent that you can install on your virtual machines, which reports any security vulnerabilities.
* Certificate Manager?—?The service offers free SSL certificates for your domains that are managed by Route53.
* WAF (Web Application Firewall)?— WAF security service offers application-level protection and allows you to block SQL injection and helps you to block cross-site scripting attacks.
* Cloud Directory?—?This service allows you to create flexible, cloud-native directories for managing hierarchies of data along multiple dimensions.
* KMS (Key Management Service)?—?It is a managed service. This security service helps you to create and control the encryption keys which allows you to encrypt your data.
* Organizations?—?You can create groups of AWS accounts using this service to manages security and automation settings.
* Shield?—?Shield is managed DDoS (Distributed Denial of Service protection service). It offers safeguards against web applications running on AWS.
* Macie?—?It offers a data visibility security service which helps classify and protect your sensitive critical content.
* GuardDuty —It offers threat detection to protect your AWS accounts and workloads.

**3.1.6 AWS S3**

Although Amazon Web Services (AWS) does not publicly provide the details of S3's technical design, Amazon S3 manages data with an object storage architecture which aims to provide scalability, high availability, and low latency with 99.999999999% durability and between 99.95% to 99.99% availability (though there is no service-level agreement for durability).Amazon S3 is an incredibly simple cloud service to use, but adequately securing your S3 resources is anything but simple, as too many organizations have discovered.

The basic storage units of Amazon S3 are objects which are organized into buckets. Each object is identified by a unique, user-assigned key. Buckets can be managed using either the console provided by Amazon S3, programmatically using the AWS SDK, or with the Amazon S3 REST application programming interface (API). Objects can be managed using the AWS SDK or with the Amazon S3 REST API and can be up to five terabytes in size with two kilobytes of metadata.[8][9] Additionally, objects can be downloaded using the HTTP GET interface and the BitTorrent protocol.

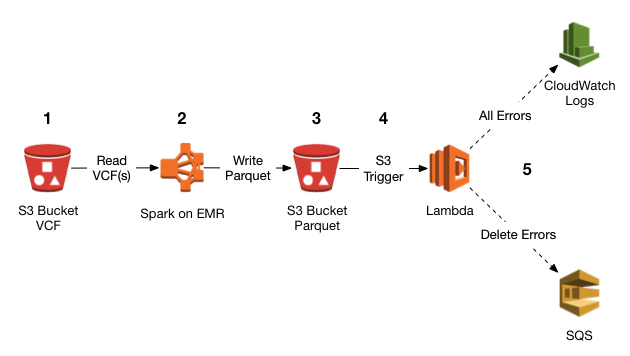
Requests are authorized using an access control list associated with each object bucket and support versioning which is disabled by default

Amazon S3 can be used to replace significant existing (static) web-hosting infrastructure with HTTP client accessible objects. The Amazon AWS authentication mechanism allows the bucket owner to create an authenticated URL which is valid for a specified amount of time.

Every item in a bucket can also be served as a BitTorrent feed. The Amazon S3 store can act as a seed host for a torrent and any BitTorrent client can retrieve the file. This can drastically reduce the bandwidth cost for the download of popular objects. While the use of BitTorrent does reduce bandwidth, AWS does not provide native bandwidth limiting and, as such, users have no access to automated cost control. This can lead to users on the free-tier of Amazon S3, or small hobby users, amassing dramatic bills. AWS representatives have stated that a bandwidth limiting feature was on the design table from 2006 to 2010,but in 2011 the feature is no longer in development.

A bucket can be configured to save HTTP log information to a sibling bucket; this can be used in data mining operations.

There are various User Mode File System (FUSE)-based file systems for Unix-like operating systems (Linux, etc.) that can be used to mount an S3 bucket as a file system such as S3QL. The semantics of the Amazon S3 file system are not that of a POSIX file system, so the file system may not behave entirely as expected.The broad adoption of Amazon S3 and related tooling has given rise to competing services based on the S3 API. These services use the standard programming interface; however, they are differentiated by their underlying technologies and supporting business models.



**Fig 3.1 AWS S3(Simple Storage Service)**

A user creates a bucket. When this bucket is created, the user will specify the region in which the bucket is deployed. Later, when files are uploaded to the bucket, the user will determine the type of S3 storage class to be used for those specific objects. After this, users can define features to the bucket, such as bucket policy, lifecycle policies, versioning control, etc.A common architectural pattern is for an organization to use a number of separate AWS accounts to isolate different teams and environments. For example, a "staging" system will often be deployed into a separate AWS account than its corresponding "production" system, to minimize the risk of the staging environment affecting production infrastructure, whether via rate limiting, misconfigured access controls, or other unintended interactions.

The S3 backend can be used in a number of different ways that make different tradeoffs between convenience, security, and isolation in such an organization. This section describes one such approach that aims to find a good compromise between these tradeoffs, allowing use of Terraform's workspaces feature to switch conveniently between multiple isolated deployments of the same configuration.

Use this section as a starting-point for your approach, but note that you will probably need to make adjustments for the unique standards and regulations that apply to your organization. You will also need to make some adjustments to this approach to account for existing practices within your organization, if for example other tools have previously been used to manage infrastructure.

Terraform is an administrative tool that manages your infrastructure, and so ideally the infrastructure that is used by Terraform should exist outside of the infrastructure that Terraform manages. This can be achieved by creating a separate administrative AWS account which contains the user accounts used by human operators and any infrastructure and tools used to manage the other accounts. Isolating shared administrative tools from your main environments has a number of advantages, such as avoiding accidentally damaging the administrative infrastructure while changing the target infrastructure, and reducing the risk that an attacker might abuse production infrastructure to gain access to the (usually more privileged) administrative infrastructure.

**Buckets**

A bucket is a container used for storing the objects.

Every object is incorporated in a bucket.

A bucket has no limit to the amount of objects that it can store. No bucket can exist inside of other buckets.

S3 performance remains the same regardless of how many buckets have been created.

The AWS user that creates a bucket owns it, and no other AWS user cannot own it. Therefore, we can say that the ownership of a bucket is not transferrable.

The AWS account that creates a bucket can delete a bucket, but no other AWS user can delete the bucket.

**Objects**

Objects are the entities which are stored in an S3 bucket.

An object consists of object data and metadata where metadata is a set of name-value pair that describes the data.

An object consists of some default metadata such as date last modified, and standard HTTP metadata, such as Content type. Custom metadata can also be specified at the time of storing an object.

It is uniquely identified within a bucket by key and version ID.

**Key**

A key is a unique identifier for an object.

Every object in a bucket is associated with one key.

An object can be uniquely identified by using a combination of bucket name, the key, and optionally version ID.

For example, in the URL http://jtp.s3.amazonaws.com/2019-01-31/Amazons3.wsdl where "jtp" is the bucket name, and key is "2019-01-31/Amazons3.wsdl"

**Regions**

You can choose a geographical region in which you want to store the buckets that you have created.

A region is chosen in such a way that it optimizes the latency, minimize costs or address regulatory requirements.

Objects will not leave the region unless you explicitly transfer the objects to another region.

Data Consistency Model

Amazon S3 replicates the data to multiple servers to achieve high availability.

Two types of model:

Read-after-write consistency for PUTS of new objects.

For a PUT request, S3 stores the data across multiple servers to achieve high availability.

A process stores an object to S3 and will be immediately available to read the object.

A process stores a new object to S3, it will immediately list the keys within the bucket.

It does not take time for propagation, the changes are reflected immediately.

Eventual consistency for overwrite PUTS and DELETES

For PUTS and DELETES to objects, the changes are reflected eventually, and they are not available immediately.

If the process replaces an existing object with the new object, you try to read it immediately. Until the change is fully propagated, the S3 might return prior data.

If the process deletes an existing object, immediately try to read it. Until the change is fully propagated, the S3 might return the deleted data.

If the process deletes an existing object, immediately list all the keys within the bucket. Until the change is fully propagated, the S3 might return the list of the deleted key.

**Amazon S3 Storage Classes**

Let’s have a look at the different storage classes using the example of a school:

* Amazon S3 Standard for frequent data access: Suitable for a use case where the latency should below. Example: Frequently accessed data will be the data of students’ attendance, which should be retrieved quickly.
* Amazon S3 Standard for infrequent data access: Can be used where the data is long-lived and less frequently accessed. Example: Students’ academic records will not be needed daily, but if they have any requirement, their details should be retrieved quickly.
* Amazon Glacier: Can be used where the data has to be archived, and high performance is not required. Example: Ex-student’s old record (like admission fee) will not be needed daily, and even if it is necessary, low latency is not required.
* One Zone-IA Storage Class: It can be used where the data is infrequently accessed and stored in a single region. Example: Student’s report card is not used daily and stored in a single availability region (i.e., school).
* Amazon S3 Standard Reduced Redundancy storage: Suitable for a use case where the data is non-critical and reproduced quickly. Example: Books in the library are non-critical data and can be replaced if lost.

In lifecycle management, Amazon S3 applies a set of rules that define the action to a group of objects. You can manage and store objects in a cost-effective manner. There are two types of actions:

Transition Action

With this action, you can choose to move objects to another storage class. With this, you can configure S3 to move your data between various storage classes on a defined schedule. Assume you’ve got some data stored in the S3 standard class. If this data is not used frequently for 30 days, it would be moved to the S3 infrequent access class. And after 60 days, it is moved to Glacier. This helps you to migrate your data to lower-cost storage as it ages automatically.

Bucket Policy

Bucket policy is an IAM policy where you can allow or deny permission to your Amazon S3 resources. With bucket policy, you also define security rules that apply to more than one file within a bucket. For example: If you do not want a user to access the “Simplilearn” bucket, then with the help of JSON script, you can set permissions. As a result, a user would be denied access to the bucket.

An important thing to consider when deciding where to declare specifics like resource names is that you want to avoid proliferating IAM Policies because keeping track of them becomes a source of both cost and risk. Therefore, embedding specific Resource identifiers in an IAM Policy is generally undesirable as it will lead to numerous copies of the policy. We will use S3 Bucket Policies to limit resource access at the bucket layer next, so that when deployed to a user or process, this policy doesn’t allow access to all S3 buckets.

**3.1.7 IAM (Identity Access Management)**

Identity and access management (IAM) in enterprise IT is about defining and managing the roles and access privileges of individual network entities (users and devices) to a variety of cloud and on-premises applications. Users include customers, partners and employees; devices include computers, smartphones, routers, servers, controllers and sensors. The core objective of IAM systems is one digital identity per individual or item. Once that digital identity has been established, it must be maintained, modified and monitored throughout each user’s or device’s access lifecycle. Thus, the overarching goal of identity management is to grant access to the enterprise assets that users and devices have rights to in a given context. That includes onboarding users and systems, permission authorizations, and the offboarding of users and devices in a timely manner.

However, part of the problem are the users and their love/hate affair with their passwords. We all have too many passwords, making the temptation to share them across logins – and the resulting security implications – an issue. A Forrester survey from August 2020 found that 53% of information workers store their passwords insecurely. Another March 2021 survey of US consumers by Transmit Security found that more than half of them stopped using a website because their login process was too complex. Clearly there is work still to be done in this area.

IAM systems provide administrators with the tools and technologies to change a user’s role, track user activities, create reports on those activities, and enforce policies on an ongoing basis. These systems are designed to provide a means of administering user access across an entire enterprise and to ensure compliance with corporate policies and government regulations.

“Identity has become more important since COVID has made physical boundaries irrelevant,” says Andras Cser, VP and IAM analyst with Forrester Research. More businesses have moved toward remote users and have also given users outside the organization greater access to their internal systems. “With digital transformation accelerating, identity has become the cornerstone of customer acquisition, management, and retention,” he says. COVID-caused disruption has surfaced weaknesses in many organizations’ IAM architecture and greatly accelerated IAM evolution, according to Gartner’s latest 2021 Planning Guide for IAM report. “The economy now relies on IAM.”

This may be why IAM spending is up. According to a March 2021 study of more than 1,300 executives sponsored by Ping Identity, about “70% of global business executives plan to increase spending on IAM for their workforce over the next 12 months, as a continuation of remote work increases demand on IT and security teams.” They also found that more than half of the companies surveyed have invested in new IAM products since the pandemic began.

Regulating user access has traditionally involved authentication methods for verifying a user’s or device’s identity, including passwords, digital certificates, hardware and smartphone software tokens. These latter forms of tokens first emerged in 2005 and now can be found on both iOS and Android smartphones with apps from Google, Microsoft, Cisco/Duo, Authy and numerous other IAM vendors. More modern approaches include biometric elements and support for the Fast Identity Alliance (FIDO).

In today’s complex compute environments, along with heightened security threats, a strong username and password doesn’t cut it anymore. The most notable change has been the addition of multi-factor authentication (MFA) into IAM products. Today, identity management systems often incorporate elements of biometrics, machine learning and artificial intelligence, and risk-based authentication.

IAM's role in the organization’s security stack

IAM plays a series of critical roles at several places in an organization’s security “stack,” but it isn’t often thought of that way because these roles are spread out across different groups, such as development teams, IT infrastructure, operations managers, the legal department and so forth. “IAM teams are no longer making all the related decisions about IAM,” said Gartner in its planning guide.

First, IAM techniques are just the beginning of managing a secure network. They requires companies to define their access policies, specifically outlining who has access to which data resources and applications and under which conditions they have access.

Many companies have evolved their access control policies over time, and the result is that they have overlapping rules and role definitions that are usually outdated and, in some cases, provisioned incorrectly. “You have to clean up your identities and revoke all the extra privileges that users don’t need so that you don’t migrate a mess,” says Cser. “This means spending more time on upfront design.”

Second, IAM has to connect with all parts of the business, such as integration with analytics, business intelligence, customer and partner portals, and marketing solutions. “Otherwise, IAM quickly becomes irrelevant,” says Cser. Gartner recommends that IAM adopt the same continuous value delivery model that many DevOps cloud teams use to deliver their software. That isn’t how many enterprise IT shops have approached IAM in the past, however.

Next, IAM goes beyond protecting users to include authenticating non-human entities such as application keys, APIs, and secrets, agents and containers. Gartner recommends making these items “first-class citizens” and says they should be managed appropriately with cross-functional teams to bring together every stakeholder. This is one area where IAM is evolving rapidly, as evidenced by the acquisition of Auth0 by Okta earlier this year.

Finally, IAM needs to be tied closely with adaptive authentication and MFA tools. Authentication used to be thought of as a binary go/no-go decision at the moment of login, such as signing into a VPN. That’s old-world thinking. Today’s IAM needs more granularity to prevent account takeovers and subtle phishing attacks. Gartner recommends rolling out adaptive MFA to all users and having an evolving authorization model that safely enables remote access. This both increases trust and improves overall usability, and as Gartner’s planning guide states, “adaptive access is just the beginning of smarter authentication solutions. Most of these products don’t have fraud detection based on passive biometric collections or support digital signatures and identity orchestrations. These protections that are needed thanks to new and more sophisticated account takeover attack methods.”

**What IAM means for compliance**

IAM systems can bolster regulatory compliance by providing the tools to implement comprehensive security, audit and access policies. Many systems now provide features designed to ensure that an organization is in compliance.

Many governments require enterprises to care about identity management. Regulations such as Sarbanes-Oxley, Gramm-Leach-Bliley and HIPAA hold organizations accountable for controlling access to customer and employee information. Identity management systems can help organizations comply with those regulations.

The General Data Protection Regulation (GDPR) requires strong security and user access controls. GDPR mandates that organizations safeguard the personal data and privacy of European Union citizens and businesses. Various US states have enacted similar privacy laws. To comply with these laws means you need to automate many aspects of IAM, and to ensure that your workflows, processes, access rights, and applications stay in compliance.

IAM open standards

The good and bad news about IAM is that there are numerous open standards to track and to leverage. These standards are a great starting point, but as Gartner mentions in its planning guide, organizations need to go beyond embracing particular open standards and be more nuanced about how to adopt these standards and be more effective at managing access. “For example, the IAM team should develop best practice documents on how these standards are integrated and used across all applications, devices, and users,” the guide said.

Authorization messages between trusted partners are often sent using Security Assertion Markup Language (SAML). This open specification defines an XML framework for exchanging security assertions among security authorities. SAML achieves interoperability across different vendor platforms that provide authentication and authorization services. SAML isn’t the only open-standard identity protocol, however. Others include OpenID, Web Services Trust (WS-Trust) and WS-Federation (which has corporate backing from Microsoft and IBM), and OAuth, which let a user’s account information be used by third-party services such as Facebook without exposing the password.

The biggest change in identity standards since 2013 has been the adoption of FIDO among a variety of IAM vendors, device makers and operating systems. It provides approaches for eliminating passwords entirely, using a variety of hardware security keys, biometric methods and smartphone profiles.

**What are the challenges and risks of implementing IAM?**

Despite IAM’s presence up and down an organization’s security stack, it doesn’t cover everything. One issue is how users’ “birthright access” policies evolve. These are the access rights that are given to new users when they begin working at a company. The options for how new employees, contractors, and partners are granted this access touch on numerous different departments, and “delegating this to the right people and managers becomes an issue,” says Cser. “IAM systems should be able to detect access rights changes automatically, but they often don’t.”

This level of automation becomes important, particularly if we consider automated on and offboarding of users, user self-service, and continuous proof of compliance, Steve Brasen, research director at EMA, wrote in a blog post. Manually adjusting access privileges and controls for hundreds or thousands of users isn’t feasible. For example, not having automated “leaving” processes (and auditing them periodically) will almost guarantee that unneeded access rights haven’t been completely revoked.

You can’t do this with Excel spreadsheets or other manual methods,” says Cser, “but underlying complexity of user onboarding hasn’t gotten any better over time, even as IAM products have gotten better at handling workflows and business processes.”

Second, while zero trust networks are all the rage right now, the issue is being able to continuously monitor these trust relationships as new applications are added to a corporation’s infrastructure. “We need to watch what people are doing after they login and look at behavior baselines. There are lots of false positive opportunities, such as if a user broke their finger,” that can mess up these trust relationships, says Cser.

Next, the relationship of IAM and single-sign on (SSO) needs to be carefully orchestrated. According to Gartner, “The goal is to get to one integrated SSO system per user constituency that can mediate access to all of the generations of applications the organization uses. Note that this doesn’t necessarily mean using one SSO tool across the entire organization.”

Next, the grand unification of IAM with customer-centric IAM has begun, as witnessed by Okta’s Auth0 acquisition. As long as these are seen as two separate efforts by security professionals, IAM will always be playing catch-up.

Next, IAM teams need to be conversant with multiple cloud architectures. See examples of IAM security best practices for Amazon Web Services (AWS), Google Cloud Platform and Microsoft Azure. Integrating these practices with an organization’s network and applications infrastructure will be challenging and bridging the security gaps among these cloud providers won’t be easy.

Finally, IT managers need to build in identity management from the start with any new applications. Cser suggests carefully selecting a target app that can be used as a template to pilot any IAM and identity governance and then expand to other apps across the enterprise.

**IAM Terms**

Buzzwords come and go, but a few key terms in the identity management space are worth knowing:

Access management: Access management refers to the processes and technologies used to control and monitor network access. Access management features, such as authentication, authorization, trust and security auditing, are part and parcel of the top ID management systems for both on-premises and cloud-based systems.

Active Directory (AD): Microsoft developed AD as a user-identity directory service for Windows domain networks. Though proprietary, AD is included in the Windows Server operating system and is thus widely deployed.

Biometric authentication: A security process for authenticating users that relies upon the user’s unique characteristics. Biometric authentication technologies include fingerprint sensors, iris and retina scanning, and facial recognition.

Context-aware network access control: Context-aware network access control is a policy-based method of granting access to network resources according to the current context of the user seeking access. For example, a user attempting to authenticate from an IP address that hasn’t been whitelisted would be blocked.

Credential: An identifier employed by the user to gain access to a network such as the user’s password, public key infrastructure (PKI) certificate, or biometric information (fingerprint, iris scan).

De-provisioning: The process of removing an identity from an ID repository and terminating access privileges.

Digital identity: The ID itself, including the description of the user and his/her/its access privileges. (“Its” because an endpoint, such as a laptop or smartphone, can have its own digital identity.)

Entitlement: The set of attributes that specify the access rights and privileges of an authenticated security principal.

Identity as a Service (IDaaS): Cloud-based IDaaS offers identity and access management functionality to an organization’s systems that reside on-premises and/or in the cloud.

Identity lifecycle management: Similar to access lifecycle management, the term refers to the entire set of processes and technologies for maintaining and updating digital identities. Identity lifecycle management includes identity synchronization, provisioning, de-provisioning, and the ongoing management of user attributes, credentials and entitlements.

Identity synchronization: The process of ensuring that multiple identity stores—say, the result of an acquisition—contain consistent data for a given digital ID.

Lightweight Directory Access Protocol (LDAP): LDAP is open standards-based protocol for managing and accessing a distributed directory service, such as Microsoft’s AD

Multi-factor authentication (MFA): MFA is when more than just a single factor, such as a user name and password, is required for authentication to a network or system. At least one additional step is also required, such as receiving a code sent via SMS to a smartphone, inserting a smart card or USB stick, or satisfying a biometric authentication requirement, such as a fingerprint scan.

Password reset: In this context, it’s a feature of an ID management system that allows users to re-establish their own passwords, relieving the administrators of the job and cutting support calls. The reset application is often accessed by the user through a browser. The application asks for a secret word or a set of questions to verify the user’s identity.

Privileged account management: This term refers to managing and auditing accounts and data access based on the privileges of the user. In general terms, because of his or her job or function, a privileged user has been granted administrative access to systems. A privileged user, for example, would be able set up and delete user accounts and roles.Provisioning: The process of creating identities, defining their access privileges and adding them to an ID repository.

Risk-based authentication (RBA): Risk-based authentication dynamically adjusts authentication requirements based on the user’s situation at the moment authentication is attempted. For example, when users attempt to authenticate from a geographic location or IP address not previously associated with them, those users may face additional authentication requirements.

Security principal: A digital identity with one or more credentials that can be authenticated and authorized to interact with the network.

Single sign-on (SSO): A type of access control for multiple related but separate systems. With a single username and password, a user can access a system or systems without using different credentials.

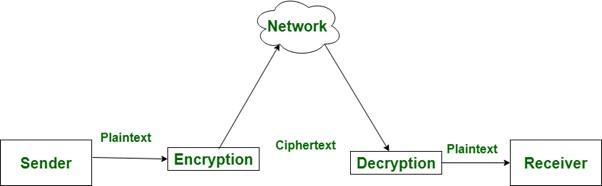
User behavior analytics (UBA): UBA technologies examine patterns of user behavior and automatically apply algorithms and analysis to detect important anomalies that may indicate potential security threats. UBA differs from other security technologies, which focus on tracking devices or security events. UBA is also sometimes grouped with entity behavior analytics and known as UEBA. In a simple implementation of the pattern described in the prior sections, all users have access to read and write states for all workspaces. In many cases it is desirable to apply more precise access constraints to the Terraform state objects in S3, so that for example only trusted administrators are allowed to modify the production state, or to control reading of a state that contains sensitive information.

Amazon S3 supports fine-grained access control on a per-object-path basis using IAM policy. A full description of S3's access control mechanism is beyond the scope of this guide, but an example IAM policy granting access to only a single state object within an S3 bucket is shown below:

## CRYPTOGRAPHY

Cryptography is the procedure of getting data and interchanges through utilization of codes so just those individuals for whom the data is proposed can get it and interaction it. In this way, forestalling unapproved admittance to data. The prefix "sepulcher" signifies "covered up" and the postfix chart signifies "composing".

In Cryptography the strategies which are utilized to shield data are acquired from numerical ideas and a bunch of rule-based estimations known as calculations to change over messages in manners that make it difficult to disentangle it. These calculations are utilized for cryptographic key age, advanced marking, confirmation to ensure information security, web perusing on the web and to secure classified exchanges, for example, Mastercard and check card exchanges [14].



**Fig 3.2 Cryptography Building Blocks**

* + 1. **Techniques used in Cryptography:**

Highlights of Cryptography are as per the following:

1. Privacy: Information must be gotten to by the individual for whom it is expected and no other individual aside from him can get to it.
2. Trustworthiness: Information can't be adjusted away or change among sender and proposed recipient with no expansion to data being distinguished.
3. Non-renouncement: The maker/sender of data can't deny their aim to send data at a later stage.
4. Validation: The characters of sender and beneficiary are affirmed. Just as objective/beginning of data is affirmed.

A cryptosystem is an execution of cryptographic procedures and their going with foundation to give data security administrations. A cryptosystem is additionally alluded to as a code framework.

### Components of a Cryptosystem

The various components of a basic cryptosystem are as follows −

* + - 1. Plaintext: It is the data to be protected during transmission.
      2. Encryption Algorithm: It is a mathematical process that produces a ciphertext for any given plaintext and encryption key. It is a cryptographic algorithm that takes plaintext and an encryption key as input and produces a ciphertext.
      3. Ciphertext: It is the scrambled version of the plaintext produced by the encryption algorithm using a specific encryption key. The ciphertext is not guarded. It flows on a public channel. It can be intercepted or compromised by anyone who has access to the communication channel.
      4. Decryption Algorithm: It is a mathematical process that produces a unique plaintext for any given ciphertext and decryption key. It is a cryptographic algorithm that takes a ciphertext and a decryption key as input, and outputs a plaintext. The decryption algorithm essentially reverses the encryption algorithm and is thus closely related to it.
      5. Encryption Key: It is a value that is known to the sender. The sender inputs the encryption key into the encryption algorithm along with the plaintext in order to compute the ciphertext.
      6. Decryption Key: It is a value that is known to the receiver. The decryption key is related to the encryption key, but is not always identical to it. The receiver inputs the decryption key into the decryption algorithm along with the ciphertext in order to compute the plaintext.

### Encryption

In the cutting edge world, Cryptography vigorously depends upon subjects like arithmetic and software engineering. Calculations for Cryptography are planned so that they are difficult to break practically speaking by any malevolent outsider otherwise called foes. A down to earth approach toward breaking such a calculation would come up short, notwithstanding, the hypothetical methodology may potentially break such a framework. Consequently, any calculation can be refered to as secure, if its key properties can't be found, with a given ciphertext.

Encryption is one of the segments of the Cryptography, which is the best and famous information security method. Encryption measure includes changing the information into another structure, known as ciphertext, though unique information to be scrambled is known as plaintext. Plaintext is provided to a calculation and an encryption key, which make a ciphertext. This ciphertext can be unscrambled with a legitimate key. Information which is put away on the PC should be moved utilizing the web or PC organization. While sending the information across an organization, uprightness or security of computerized information should be kept up, encryption assumes a vital part in giving information trustworthiness. There are some center protections highlights which should be kept up: information trustworthiness, verification, and non-disavowal. Verification implies the information's starting point should be checked. Information respectability would guarantee that substance isn't changed since it was being sent and, non-renouncement would guarantee the sender can't reject about sending the message. These essential security viewpoints are being served by an encryption interaction. Like Cryptography, Encryption has two modes: symmetric and lopsided. An equivalent mystery key is divided among the sender and beneficiary while performing encryption and unscrambling. The hilter kilter approach, then again, utilizes two unique keys, public and private. Encryption strategy is normal among the utilization of securing data with the regular citizen framework, by governments and military. Client's own and banking related information is exceptionally inclined to burglary, encoding such records is consistently a shelter, in the event that the security framework neglects to ensure the classified information. Encryption from the start may appear to be a confounded methodology yet different information misfortune avoidance programming handles it effectively.

### Symmetric Encryption

The symmetric encryption technique, as the name infers, utilizes a solitary cryptographic key to encode and decode information. The utilization of a solitary key for the two activities makes it a direct interaction, and henceforth it is called symmetric.Symmetric key encryption is important for the public key foundation (PKI) environment, which makes it conceivable to impart safely across the shaky web by changing over plain content (coherent) information into unrecognizable ciphertext. Indeed, you're utilizing PKI-based innovations and cycles right now as a component of your association with TheSSLStore.com to peruse this article.Symmetric calculations are separated into two fundamental sorts: stream and square codes. Square codes encode information in pieces (blocks), while stream figures scramble information the slightest bit at a time.

Information Encryption Standard (DES) — DES is a sort of square code that scrambles information in 64-bit squares and utilizing a solitary key that is one of three sizes (64-digit, 128-cycle and 192-piece keys). Notwithstanding, one of each 8 pieces is an equality bit, implying that a solitary length key that is 64 pieces is truly similar to utilizing a 56-digit key. In spite of the fact that DES is one of the soonest symmetric encryption calculations, it's seen as unreliable and has been deplored.

Triple Data Encryption Standard (TDEA/3DES) — Unlike DES, triple DES can utilize a few keys, which empowers this calculation to utilize various rounds of encryption (or, more exact, a series of encryption, round of unscrambling, and another round of encryption). While 3DES is safer than its DES archetype, it's not as secure as its replacement, AES.

Progressed Encryption Standard (AES) — This encryption calculation is the thing that you'll most regularly discover is use across the web. The high level encryption standard is safer and productive than DES and 3DES with key choices that are 128 pieces, 192 pieces and 256 pieces. Notwithstanding, while it's additionally a sort of square code, it works uniquely in contrast to DES and 3DES in light of the fact that it depends on a replacement stage network rather than the Feistel figure.

Symmetric encryption calculations aren't the solitary calculations out there that PKI relies on. There are additionally deviated calculations and uneven key trade conventions.

The present encryption strategies are not actually that straightforward. The broadly utilized encryption calculations are perplexing to the point that even the consolidated figuring force of some super-PCs can't break them.

The most outstanding feature of symmetric encryption is the simplicity of its process. This simplicity of this type of encryption lies in the use of a single key for both encryption as well as decryption. As a result, symmetric encryption algorithms:

* + - * 1. Are significantly faster than their asymmetric encryption counterparts
        2. Require less computational power, and
        3. Don’t dampen internet speed.

This means that when there’s a large chunk of data to be encrypted, symmetric encryption proves to be a great option.

The encryption methods that are used today rely on highly complex mathematical functions that make it virtually impossible to crack them.

The three common types of Symmetric Encryption Algorithms are:

**DES Symmetric Encryption Algorithm**:

DES (information encryption standard) is one of the most seasoned symmetric encryption techniques. It was created by IBM to ensure touchy, unclassified electronic government information and was officially embraced in 1977 for use by administrative offices. DES utilizes a 56-digit encryption key, and it depends on the Feistel Structure that was planned by a cryptographer named Horst Feistel. The DES encryption calculation was among those that were remembered for TLS (transport layer security) adaptations 1.0 and 1.1.

DES changes over 64-bit squares of plaintext information into ciphertext by partitioning the square into two separate 32-bit obstructs and applying the encryption cycle to each freely. This includes 16 rounds of different cycles like development, stage, replacement, OR, AND, XOR activity with a round key that the information will go through as it's scrambled. At last, 64-cycle squares of encoded text is delivered as the yield.

DES is not, at this point being used as it was broken by numerous security scientists. In 2005, DES was authoritatively deplored and was supplanted by the AES encryption algorithm.The greatest disadvantage to DES was its low encryption key length, which made animal driving simple against it. TLS 1.2, the most generally utilized TLS convention today, doesn't utilize the DES encryption technique.

3DES Symmetric Encryption Algorithm:

3DES (otherwise called TDEA, which represents triple information encryption calculation), as the name suggests, is an overhauled adaptation of the DES calculation that was delivered. 3DES was created to conquer the disadvantages of the DES calculation and was placed into utilization beginning in the last part of the 1990s. To do as such, it applies the DES calculation threefold to every information block. Thus, this interaction made 3DES a lot harder to break than its DES archetype. It additionally turned into a generally utilized encryption calculation in installment frameworks, guidelines, and innovation in the account business. It's additionally become a piece of cryptographic conventions like TLS, SSH, IPsec, and OpenVPN.

All encryption calculations at last surrender to the force of time, and 3DES was the same

Taking over from the traditional DES component, triple DES was at present carried out in the security draws near. These calculations license programmers to at last acquire the information to defeat in a simple methodology. This was the widely carried out approach by numerous individuals of the endeavors. Triple DES works with 3 keys having 56 pieces for every each key. The whole key length is a limit of pieces, though specialists would battle that 112-bits in key force is more plausible. This calculation handles a solid equipment encryption answer for banking offices and furthermore for different enterprises.

AES Symmetric Encryption Algorithm:

AES, which means "progressed encryption framework," is perhaps the most predominant kinds of encryption calculations and was created as an option in contrast to the DES calculation. AES turned into an encryption standard on endorsement by NIST in 2001. Not at all like DES, AES is a group of square codes that comprises of codes of various key lengths and square sizes.

AES deals with the techniques for replacement and stage. To begin with, the plaintext information is transformed into squares, and afterward the encryption is applied utilizing the encryption key. The encryption cycle comprises of different sub-cycles like sub bytes, shift lines, blend sections, and add round keys. Contingent on the size of the key, 10, 12, or 14 such adjusts are performed. It's important that the last round does exclude the sub-interaction of blend segments among any remaining sub-measures performed to encode the information.

This is the most confided in calculation strategy by the U.S organization and numerous different endeavors. Despite the fact that this works productively in 128-bit encryption structure, 192 and 256 pieces are chiefly utilized for colossal encryption exercises. Being so safe to all hacking frameworks, the AES strategy gets broad commendation for scrambling data in the private area.

The Advantage of Using the AES Encryption Algorithm is that AES is protected, quick, and adaptable. AES is a lot faster calculation contrasted with DES. The numerous key length alternatives are the greatest benefit we have as the more drawn out the keys are, the harder it is to break them.

Today, AES is the most generally utilized encryption calculation — it's utilized in numerous applications, including:

1. Remote security
2. Processor security and document encryption,
3. SSL/TLS convention (site security),
4. Wi-Fi security,
5. Portable application encryption,
6. VPN (virtual private organization), and so forth

Numerous administration offices, including the National Security Agency (NSA), depend on the AES encryption calculation to ensure their delicate data.

### Asymmetric Encryption

Topsy-turvy encryption, as opposed to the symmetric encryption technique, includes various keys for encryption and decoding of the information. Asymmetric encryption utilizes two particular, yet related, keys. One key, the Public Key, is utilized for encryption, and the other, the Private Key, is for decoding. As suggested in the name, the Private Key is proposed to be private with the goal that solitary the confirmed beneficiary can decode the message. Under this framework a couple of keys is utilized to encode and unscramble data. A public key is utilized for encryption and a private key is utilized for decoding. Public key and Private Key are unique. Regardless of whether the public key is known by everybody the proposed recipient can just translate it since he alone knows the private key.

The main benefit of this sort of encryption is the security it gives. In this strategy, the public key-which is freely accessible, is utilized to encode the information, while the decoding of the information is finished utilizing the private key, which should be put away safely. This guarantees that the information stays secured against man-in-the-center (MiTM) assaults. For web/email workers that associate with countless customers ever minute, uneven encryption isn't anything not exactly a help as they just need to oversee and secure a solitary key. Another central issue is that public key cryptography permits making a scrambled association without meeting disconnected to trade keys first.

The second significant component that uneven encryption offers is confirmation. The information encoded by a public key must be unscrambled utilizing the private key identified with it. Accordingly, it ensures that the information is just seen and unscrambled by the element that should get it.

### 3.2.3.2.1 The Two Main Types of Asymmetric Encryption Algorithms

#### RSA Asymmetric Encryption Algorithm

The RSA encryption calculation, which represents Rivest-Shamir-Adleman (the last names of the three individuals who made it), is a validation and key trade component that is usually utilized in the TLS 1.2 handshake measure. In a RSA key trade, public key encryption works with the trading of a pre-ace mystery and an arbitrarily produce number from the customer that, together, create a common meeting key.

With the Diffie-Hellman key trade, the worker and customer rather commonly concur upon a worth that is utilized for the meeting key.

One of the public-key encryption calculations used to scramble data sent through the web. It was a broadly utilized calculation in GPG and PGP strategies. RSA is arranged under uneven sort of calculation as it plays out its activity a few keys. One of the keys is utilized for encryption and the other for decoding purposes.

Imagined by Ron Rivest, Adi Shamir, and Leonard Adleman (subsequently "RSA") in 1977, RSA is, until now, the most broadly utilized unbalanced encryption calculation. Its intensity lies in the "prime factorization" strategy that it depends upon. Fundamentally, this technique includes two gigantic arbitrary indivisible numbers, and these numbers are duplicated to make another goliath number. The riddle here is to decide the first indivisible numbers from this goliath measured duplicated number.

It turns out this riddle is basically outlandish if utilizing the correct key length that is produced with sufficient entropy for the present super-PCs. In 2010, a gathering of specialists did research, and it took them over 1,500 years of figuring time (circulated across many PCs) to break the RSA-768 cycle key – which is route beneath the standard 2048-bit RSA key that is being used today.the utilization of RSA for key trades is disliked (albeit a few frameworks are as yet utilizing it) because of weaknesses that were found by cryptologist Daniel Bleichenbacher. Indeed, the RSA key trade figure suites (and non-vaporous Diffie-Hellman gatherings) were belittled with the rollout of TLS 1.3 with an end goal to order amazing forward mystery (which utilizes a fleeting key). Along these lines, RSA key trade was supplanted by the restrictive utilization of vaporous Diffie-Hellman key trades.

The Diffie-Hellman key trade calculation is a public key conveyance framework that utilizes secluded math to go to a settled upon secret number (meeting key). In this way, as should be obvious, considering Diffie-Hellman an "encryption" calculation is really deceptive on the grounds that it can't be utilized to scramble or unscramble anything. Also, it's less a key "trade" as it is a key age measure. Indeed, there are factors traded, however you're still really making the key dependent on those trades.

Diffie-Hellman utilizes the trading of public factors (numbers) to produce a common arrangement known as a meeting key. This mysterious meeting key is the thing that you'd use to trade information in a safe channel that is ensured by symmetric encryption.

An extraordinary benefit is that RSA offers is its versatility. It comes in different encryption key lengths like 768-piece, 1024-digit, 2048-piece, 4096-piece, and so forth Subsequently, regardless of whether the lower key-lengths are effectively savage constrained, It can be utilized for encryption of higher key lengths on the grounds that the trouble of beast driving the critical increments with each extending key length.

RSA depends on a basic numerical methodology, and that is the reason its execution in the public key framework (PKI) gets clear. This versatility with PKI and its security has made RSA the most generally utilized deviated encryption calculation utilized today. RSA is widely utilized in numerous applications, including SSL/TLS endorsements, digital forms of money, and email encryption

#### ECC Asymmetric Encryption Algorithm

In 1985, two mathematicians named Neal Koblitz and Victor S. Miller proposed the use of elliptic curves in cryptography. After almost two decades, their idea was turned into a reality when ECC (Elliptic Curve Cryptography) algorithm entered into use in 2004-05.

In the ECC encryption process, an elliptic curve represents the set of points that satisfy a mathematical equation (y2 = x3 + ax + b).

Like RSA, ECC also works on the principle of irreversibility. In simpler words, it’s easy to compute it in one direction but painfully difficult to reverse it and come to the original point. In ECC, a number symbolizing a point on the curve is multiplied by another number and gives another point on the curve. Now, to crack this puzzle, we must figure out the new point on the curve. The mathematics of ECC is built in such a way that it’s virtually impossible to find out the new point, even if we know the original point.

Compared to RSA, ECC offers greater security (against current methods of cracking) as it’s quite complex. It provides a similar level of protection as RSA, but it uses much shorter key lengths. As a result, ECC applied with keys of greater lengths will take considerably more time to crack using brute force attacks.

Another advantage of the shorter keys in ECC is faster performance. Shorter keys require less networking load and computing power, and that turns out to be great for devices with limited storage and processing capabilities. When the ECC is used in SSL/TLS certificates, it decreases the time it takes to perform SSL/TLS handshakes considerably and helps us load the website faster. The ECC encryption algorithm is used for encryption applications, to apply digital signatures, in pseudo-random generators, etc.

The challenge with using ECC, though, is that many server software and control panels haven’t yet added support for ECC SSL/TLS certificates. We’re hoping that this changes in the future, but this means that RSA is going to continue to be the more widely used asymmetric encryption algorithm in the meantime.

|  |  |
| --- | --- |
| **Symmetric Encryption** | **Asymmetric Encryption** |
| A single key is used to encrypt and decrypt the data | A key pair is used for encryption and decryption. These keys are known as public and private keys. |
| As it uses only one key it’s a simpler method of encryption. | Thanks to the key pair , it’s a more complex processes. |
| Symmetric encryption is primarily used for encryption | Asymmetric Encryption ensures encryption, authentication and non- repudiation. |

|  |  |
| --- | --- |
| It provides faster performance and less computational power compared to asymmetric encryption. | It’s slower than symmetric encryption and requires higher computational power because of its complexity. |
| Smaller key lengths are used to encrypt the data. | Usually, asymmetric methods involve longer keys. |
| Ideal for applications where a large amount of data needs to be encrypted. | Standard asymmetric encryption algorithms include RSA, Diffie hillman, ECC, El Gamal and DSA. |

**Table 3.1 Symmetric encryption vs Asymmetric Encryption**

### Hybrid Encryption: Symmetric + Asymmetric Encryption

It is the technique where the best of both of the strategies (Symmetric and Asymmetric) are taken to make a collaboration to construct powerful encryption frameworks. However favorable as symmetric and unbalanced encryption seem to be, the two of them have their disadvantages. The symmetric encryption strategy turns out incredible for quick encryption of huge information. All things considered, it doesn't give character confirmation, something that is the need of great importance with regards to web security. Then again, lopsided encryption ensures that the information is gotten to by our planned beneficiary. Notwithstanding, this confirmation makes the encryption interaction horrendously lethargic when carried out at scale.[15]

### Key Differences Between Cryptography and Encryption

A portion of the significant contrasts are:

* + - 1. Cryptography is the investigation of ideas like Encryption, decoding, used to give secure correspondence while encryption is the way toward encoding a message with a calculation.
      2. Cryptography can be considered as a field of study, which includes a ton of methods and innovations though Encryption is all the more a numerical and algorithmic nature.
      3. Cryptography, being a field of study has more extensive classifications and reaches, encryption is one such method though Encryption is one of the parts of Cryptography that can encode correspondence measures proficiently.
      4. Cryptography is more conventional in nature, utilizes advanced signature and another method of strategies to give security to computerized information though Encryption is being used with a bunch of calculations broadly known as a code to scramble the advanced information.
      5. Cryptography has a symmetric and lopsided variant, with an idea of shared and non-shared key though Encryption follows similar methodology for certain particular terms like ciphertext, plaintext, and figure.
      6. Cryptography includes working with calculations with fundamental cryptographic properties though Encryption is one of the subsets of Cryptography that utilizes numerical calculations called figure.
      7. Cryptography has its application which is wide and going from computerized information to the old style cryptography while Encryption is used to encode the information on the way over a PC organization.
      8. Cryptography's fields incorporate PC programming, calculation, math, data hypothesis, transmission innovation while Encryption is a greater amount of digitalized in nature, since the advanced time.
      9. Cryptography includes two significant parts called Encryption and Decryption though Encryption is an interaction of protecting data to forestall unapproved and illicit use.
      10. Cryptography goes about as a superset of Encryption, for example each interaction and terms utilized for Encryption can be supposed to be a piece of Cryptography while Encryption being a subset has its own particular terms and cycles.

### Decryption

It is a technique of changing information which has been refined as undecipherable material through encryption to its understandable state. In this cycle, the framework gets and changes over the

befuddling information into words and pictures that are just intelligible both for the peruser and framework. It very well may be performed consequently or physically. It may even be refined with a grouping of codes or passwords.

It is a cycle to disclose the got information and for it, the design acquires and changes the stirred up information and adjusts it an open language and picture for both the peruser alongside the framework. The unscrambled information got by anybody where a window will come up to enter the secret word needed to acquire the scrambled information. It very well may be performed consequently or physically just as it very well may be even done through the assortment of passwords or codes.

The main source for executing a decoding processor encryption measure is security. It turns into a matter of investigation and openness from unapproved individuals or organizations as information moves across the World Wide Web. As an outcome, data is scrambled to decrease the misfortune and burglary of information. Not many of the typical things are scrambled involve pictures, registries; email messages, client information, and text records. The individual taking care of this gets a prompt window where it needs to punch in it to get encoded information.

**3.2.5.1 Decryption Usage**

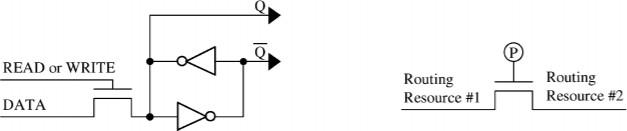
The improvement of ceaseless calculations for significant encryption has a more prominent prerequisite for insight and law requirement experts in a weapons contest in calculation. In addition, associations needing to deal with the assessments of computerized security or to recuperate lost passwords address the comparable mounting difficulty. Also, using the most exceptional approaches to decode, the prerequisite for broad calculation couldn't be stayed away from which is the justification further unscrambling Federal organizations and ISVs are accepting Frontier as their decision of ammo. Besides, the ability is to help offices to have in-house unscrambling or calculation of steganographic with Frontier. Paragon is incorporated with not many of the extraordinary business decoders to give turnkey endeavor networks that convey unscrambling on various PCs across an entirety

organization.

## RECONFIGURABLE COMPUTING

Reconfigurable computing as a concept has been in existence for quite some time. Even general-purpose processors use some of the same basic ideas, such as reusing computational components for independent computations, and using multiplexers to control the routing between

these components. However, the term reconfigurable computing, as it is used in current research , refers to systems incorporating some form of hardware programmability—customizing how the hardware is used using a number. of physical control points. These control points can then be changed periodically in order to execute different applications using the same hardware.[11]



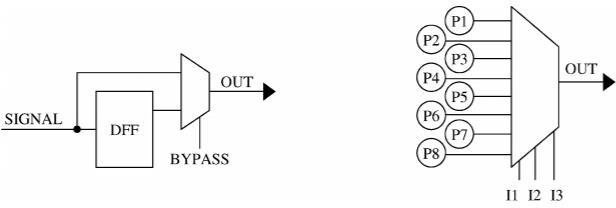
#### Fig 3.3 and Fig 3.4 A programming bit for SRAM-based FPGAs [Xilinx 1994] (left) and a programmable routing connection (right).

The recent advances in reconfigurable computing are for the most part derived from the technologies developed for FPGAs in the mid-1980s. FPGAs were originally created to serve as a hybrid device between PALs and Mask Programmable Gate Arrays (MPGAs). Like PALs, FPGAs are fully electrically programmable, meaning that the physical design costs are amortized over multiple application circuit implementations, and the hardware can be customized nearly instantaneously. Like MPGAs, they can implement very complex computations on a single chip, with devices currently in production containing the equivalent of over a million gates. Because of these features, FPGAs had been primarily viewed as glue logic replacement and rapid-prototyping vehicles. However, as we show throughout this article, the flexibility, capacity, and performance of these devices has opened up completely new avenues in high-performance computation, forming the basis of reconfigurable computing. Most current FPGAs and reconfigurable devices are SRAM- programmable, meaning that SRAM1 bits are connected to the configuration points in the FPGA, and programming the SRAM bits configures the FPGA.[11]

Thus, these chips can be programmed and reprogrammed about as easily as a standard static RAM. In fact, one research project, the PAM project, considers a group of one or more FPGAs to be a RAM unit that performs computation between the memory write (sending the configuration information and input data) and memory read (reading the results of the computation). This leads some to use the term Programmable Active Memory or PAM. One example of how the SRAM configuration points can be used is to control routing within a reconfigurable device. To configure

the routing on an FPGA, typically a pass gate structure is employed Here the programming bit will turn on a routing connection when it is configured with a true value, allowing a signal to flow from one wire to another, and will disconnect these resources when the bit is set to false. With a proper interconnection of these elements, which may include millions of routing choice points within a single device, a rich routing fabric can be created.

Another example of how these configuration bits may be used is to control multiplexers, which will choose between the output of different logic resources within the array. For example, to provide optional state holding elements a D flip-flop (DFF) may be included with a multiplexer selecting whether to forward the latched or unlatched signal value. Thus, systems that require state holding the programming bits controlling the multiplexer would be configured to select the DFF output, while systems that do not need this function would choose the bypass route that sends the input directly to the output.[11]



#### Fig 3.5 and Fig 3.6 D flip-flop with optional bypass (left) and a 3-input LUT (right)

Finally, the configuration bits may be used as control signals for a computational unit or as the basis for computation itself. As a control signal, a configuration bit may determine whether an ALU performs an addition, subtraction, or other logic computations. On the other hand, with a structure such as a lookup table (LUT), the configuration bits themselves form the result of the computation. These elements are essentially small memories provided for computing arbitrary logic functions. LUTs can compute any function of N inputs (where N is the number of control signals for the LUT’s multiplexer) by programming the 2N programming bits with the truth table of the desired function. Thus, if all programming bits except the one corresponding to the input pattern 111 were set to zero a 3-input LUT would act as a 3-input AND gate, while programming it with all ones except in 000 would compute a NAND.

## PARALLEL COMPUTING

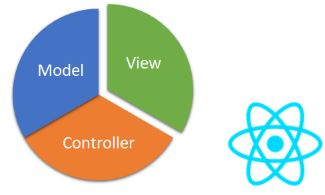
One of the benefits of reconfigurable computing is the ability to execute multiple operations in parallel. In cases where circuits are specified using a structural hardware description language, the user specifies all structures and timing, and therefore either implicitly or explicitly specifies any parallel operation. However, for behavioural and HLL descriptions, there are two methods to incorporate parallelism: manual parallelization through special instructions or compiler directives, and automatic parallelization by the Compiler. To manually incorporate parallelism within an application, the programmer can specifically mark sections of code that should run as parallel threads, and use similar operations to those used in traditional parallel compilers. For example, a signal/wait technique can be used to perform synchronization of the different threads of the computation. The RaPiD-B language is one that uses this methodology.[11]

Although the NAPA C compiler requires programmers to mark the areas of code for executing the host processor and the reconfigurable hardware in parallel, it also detects and exploits fine-grained parallelism within computations destined for the reconfigurable hardware. Automatic parallelization of inner loops is another common technique in reconfigurable hardware compilers to attempt to maximize the use of the reconfigurable hardware. The compiler will select the innermost loop level to be completely unrolled for parallel execution in hardware, potentially creating a heavily pipelined structure. For these cases, outer loops may not have multiple iterations executing simultaneously. Any loop reordering to improve the parallelism of the circuit must be done by the programmer. However, some compiler systems have taken this procedure a step further and focus on the parallelization of all loops within the program, not just the inner loops. This type of compiler generates a control flow graph based upon the entire program source code. Loop unrolling is used in order to increase the available parallelism, and the graph is then used to schedule parallel operations in the hardware.

## REACTJS

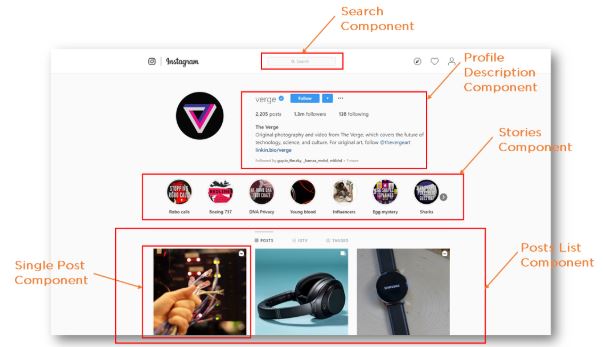
**3.5.1 What is React?**

React is a JavaScript library created for building fast and interactive user interfaces for web and mobile applications. It is an open-source, component-based, front-end library responsible only for the application’s view layer. In Model View Controller (MVC) architecture, the view layer is responsible for how the app looks and feels. React was created by Jordan Walke, a software engineer at Facebook.



**Fig:3.7 MVC architecture**

For example, Consider an Instagram webpage,It is entirely built using React.React divides the UI into multiple components, which makes the code easier to debug. This way, each component has its property and function.



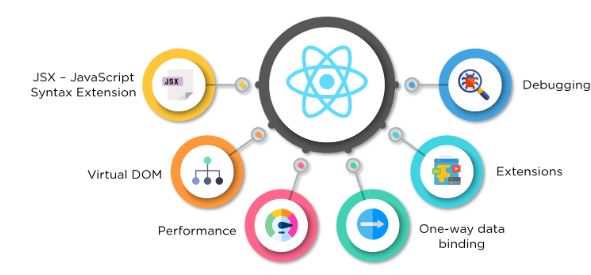
**Fig:3.8 Instagram Components**

## 3.5.2 Why React?

React’s popularity today has eclipsed that of all other front-end development frameworks.

* Easy creation of dynamic applications: React makes it easier to create dynamic web applications because it requires less coding and offers more functionality, as opposed to JavaScript, where coding often gets complex very quickly.
* Improved performance: React uses Virtual DOM, thereby creating web applications faster. Virtual DOM compares the components’ previous states and updates only the items in the Real DOM that were changed, instead of updating all of the components again, as conventional web applications do.
* Reusable components: Components are the building blocks of any React application, and a single app usually consists of multiple components. These components have their logic and controls, and they can be reused throughout the application, which in turn dramatically reduces the application’s development time.
* Unidirectional data flow: React follows a unidirectional data flow. This means that when designing a React app, developers often nest child components within parent components. Since the data flows in a single direction, it becomes easier to debug errors and know where a problem occurs in an application at the moment in question.
* Small learning curve: React is easy to learn, as it mostly combines basic HTML and JavaScript concepts with some beneficial additions. Still, as is the case with other tools and frameworks, you have to spend some time to get a proper understanding of React’s library.
* It can be used for the development of both web and mobile apps: We already know that React is used for the development of web applications, but that’s not all it can do. There is a framework called React Native, derived from React itself, that is hugely popular and is used for creating beautiful mobile applications. So, in reality, React can be used for making both web and mobile applications.
* Dedicated tools for easy debugging: Facebook has released a Chrome extension that can be used to debug React applications. This makes the process of debugging React web applications faster and easier.

## 3.5.3 Features of React



**Fig:3.9 React Features**

### JSX - JavaScript Syntax Extension

JSX is a syntax extension to JavaScript. It is used with React to describe what the user interface should look like. By using JSX, we can write HTML structures in the same file that contains JavaScript code. This makes the code easier to understand and debug, as it avoids the usage of complex JavaScript DOM structures.

The JSX code is implemented in React as neither a string nor HTML. Instead, it embeds HTML into JavaScript code.

### Virtual DOM

React keeps a lightweight representation of the “real” DOM in the memory, and that is known as the “virtual” DOM (VDOM). Manipulating real DOM is much slower than manipulating VDOM because nothing gets drawn on the screen. When the state of an object changes, VDOM changes only that object in the real DOM instead of updating all of the objects.

It may all seem a bit overwhelming for now, so let’s first understand what DOM is, and then we’ll go through how VDOM and real DOM interact with each other.

DOM (Document Object Model) treats an XML or HTML document as a tree structure in which each node is an object representing a part of the document.

When the state of an object changes in a React application, VDOM gets updated. It then compares its previous state and then updates only those objects in the real DOM instead of updating all of the objects. This makes things move fast, especially when compared to other front-end technologies that have to update each object even if only a single object changes in the web application.

### Performance

React uses VDOM, which makes the web applications run much faster than those developed with alternate front-end frameworks. React breaks a complex user interface into individual components, allowing multiple users to work on each component simultaneously, thereby speeding up the development time.

### Extensions

React goes beyond simple UI design and has many extensions that offer complete application architecture support. It provides server-side rendering, which entails rendering a normally client-side only web application on the server, and then sends a fully rendered page to the client. It also employs Flux and Redux extensively in web application development. Finally, there is React Native, a popular framework derived from React, used to create cross-compatible mobile applications.

### One-way Data Binding

React’s one-way data binding keeps everything modular and fast. A unidirectional data flow means that when a developer designs a React app, they often nest child components within parent components. This way, a developer knows where and when an error occurs, giving them better control of the whole web application.



**Fig:3.10 One-way data binding**

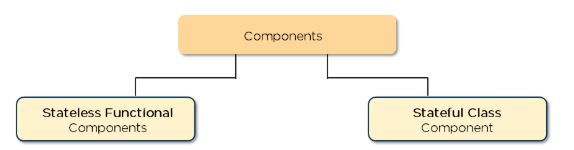
### Debugging

React applications are easy to test due to a large developer community. Facebook even provides a small browser extension that makes React debugging faster and easier.

### 3.5.4 Components

Components are the building blocks of any React application, and a single app usually consists of multiple components. A component is essentially a piece of the user interface. React splits the UI into independent, reusable parts that can be processed separately.

There are two types of components in React:



**Fig:3.11 React Components**

* Functional Components: These components have no state of their own and only contain a render method, so they are also called stateless components. They may derive data from other components as props (properties).
* Class Components: These components can hold and manage their state and have a separate render method for returning JSX on the screen. They are also called stateful components, as they can have a state.

### 3.5.5 State

The state is a built-in React object that is used to contain data or information about the component. A component’s state can change over time; whenever it changes, the component re-renders. The change in state can happen as a response to user action or system-generated events, and these changes determine the behavior of the component and how it will render.

**3.5.6 Props**

Props are short for properties. It is a React built-in object which stores the value of a tag’s attributes and works similar to the HTML attributes. It provides a way to pass data from one component to other components in the same way as arguments are passed in a function.

**Chapter 4 PROJECT APPARATUS**

**4.1 FPGA**

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks. Although one-time programmable (OTP) FPGAs are available, the dominant types are SRAM based which can be reprogrammed as the design evolves.[10]

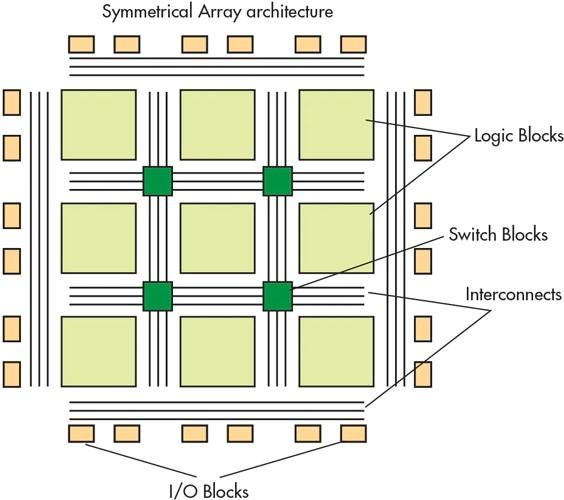
### FPGA Architecture

A precise architecture of an FPGA varies from manufacturer to manufacturer. Here, we present a generic FPGA structure that contains the following elements:

* + - 1. Programmable logic blocks: Logic blocks can be formed from thousands of transistors to millions of transistors. They implement the logic functions required by the design and consist of logic components such as transistor pairs, look-up tables (LUTs), and Carry and Control Logic (flip flops and multiplexers).
      2. Programmable I/O blocks: They connect logic blocks with external components via interfacing pins.
      3. Programmable interconnect resources: They are electrically programmable interconnections (pre-laid vertically and horizontally) that provide the routing path for the programmable logic blocks. Routing paths contain wire segments of varying lengths which can be interconnected via electrically programmable switches. The FPGA density depends on the number of segments used for routing paths.

### Types of FPGAs

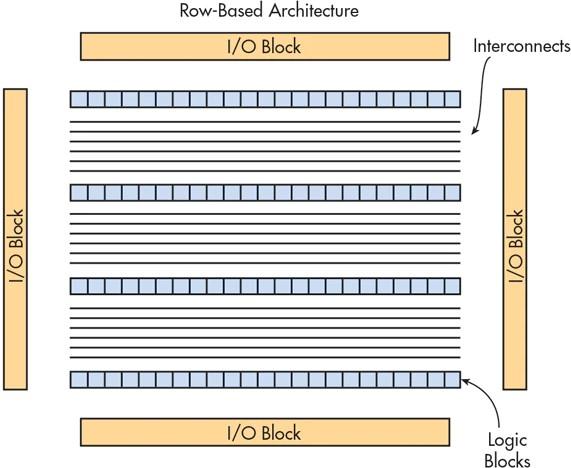
The routing architecture affects density and performance of the FPGA. Based on internal arrangement of blocks, FPGAs might be classified into three categories:

**Symmetrical arrays**: This arrangement consists of logic blocks arranged in rows and columns of a matrix and interconnect resources between them. This symmetrical matrix is surrounded by I/O blocks that connect it to the outside world.

#### Fig 4.1 Symmetrical array architecture

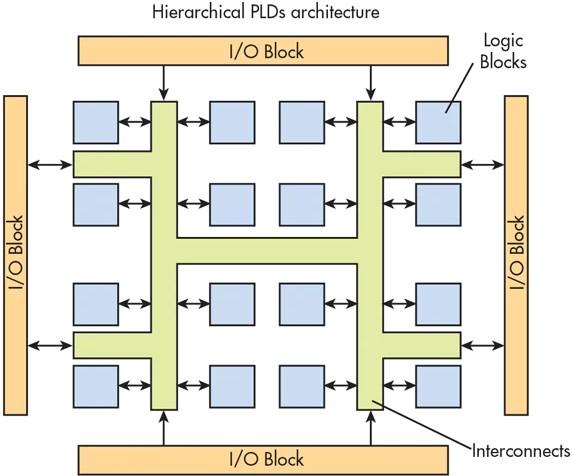
Symmetrical arrays consist of a two-dimensional array of logic modules interconnected by vertical and horizontal programmable interconnect resources.

**Row-based architecture**: It alternates rows of programmable interconnect resources with rows of logic blocks while the Input/Output blocks are located in the periphery of the rows. One row may be connected to adjacent rows via vertical interconnect.



**Fig 4.2 Row based architecture**

**Hierarchical PLDs**: These are designed in hierarchical manner with the top level containing only logic blocks and interconnects. Each logic block contains a number of logic modules. And each logic module has combinatorial as well as sequential functional elements.



#### Fig 4.3 Hierarchical Architecture

Based on programming technology type, FPGAs can be classified into three categories:

* + - 1. SRAM-based FPGAs: Static RAM cells control pass-transistor, transmission gates, or multiplexers. They can be reprogrammed as the design evolves, but when the power is off the programming is lost and they need to be configured upon start. Therefore, they need an external memory to store the program.
      2. Antifuse-based FPGAs: They use an antifuse CMOS technology and once the FPGA has been programmed, it cannot be reprogrammed. They retain their program when the power is off.
      3. Flash-based FPGAs: They use floating gate cells as switches that improve area efficiency. They do not lose information when the device is powered down. This technology does not need an external memory to store the program, but they cannot be reprogrammed infinite times due to charge build-up in the oxide.

Row-based architecture consists of rows of logic blocks that are separated by programmable interconnect resources.

* + 1. **ASIC vs FPGA**

ASIC and FPGAs have different value propositions, and they must be carefully evaluated before choosing any one over the other. Information abounds that compares the two technologies. While FPGAs used to be selected for lower speed/complexity/volume designs in the past, today’s FPGAs easily push the 500 MHz performance barrier. With unprecedented logic density increases and a host of other features, such as embedded processors, DSP blocks, clocking, and high-speed serial at ever lower price points, FPGAs are a compelling proposition for almost any type of design.[10]

### FPGA Applications

* + - 1. Due to their programmable nature, FPGAs are an ideal fit for many different markets. As the industry leader, Xilinx provides comprehensive solutions consisting of FPGA devices, advanced software, and configurable, ready-to-use IP cores for markets and applications such as:
      2. Aerospace & Defence - Radiation-tolerant FPGAs along with intellectual property for image processing, waveform generation, and partial reconfiguration for SDRs.
      3. ASIC Prototyping - ASIC prototyping with FPGAs enables fast and accurate SoC system modelling and verification of embedded software
      4. Audio - Xilinx FPGAs and targeted design platforms enable higher degrees of flexibility, faster time-to-market, and lower overall non-recurring engineering costs (NRE) for a wide range of audio, communications, and multimedia applications.
      5. Automotive - Automotive silicon and IP solutions for gateway and driver assistance systems, comfort, convenience, and in-vehicle infotainment. - Learn how Xilinx FPGAs enable Automotive Systems
      6. Broadcast & Pro AV - Adapt to changing requirements faster and lengthen product life cycles with Broadcast Targeted Design Platforms and solutions for high-end professional broadcast systems.
      7. Consumer Electronics - Cost-effective solutions enabling next generation, full-featured consumer applications, such as converged handsets, digital flat panel displays, information appliances, home networking, and residential set top boxes.
      8. Data Centre - Designed for high-bandwidth, low-latency servers, networking, and storage applications to bring higher value into cloud deployments.
      9. High Performance Computing and Data Storage - Solutions for Network Attached Storage (NAS), Storage Area Network (SAN), servers, and storage appliances.
      10. Industrial - Xilinx FPGAs and targeted design platforms for Industrial, Scientific and Medical (ISM) enable higher degrees of flexibility, faster time-to-market, and lower overall non-recurring engineering costs (NRE) for a wide range of applications such as industrial imaging and surveillance, industrial automation, and medical imaging equipment.
      11. Medical - For diagnostic, monitoring, and therapy applications, the Virtex FPGA and Spartan FPGA families can be used to meet a range of processing, display, and I/O interface requirements.
      12. Security - Xilinx offers solutions that meet the evolving needs of security applications, from access control to surveillance and safety systems.
      13. Video & Image Processing - Xilinx FPGAs and targeted design platforms enable higher degrees of flexibility, faster time-to-market, and lower overall non-recurring engineering costs (NRE) for a wide range of video and imaging applications.
      14. Wired Communications - End-to-end solutions for the Reprogrammable Networking Line Card Packet Processing, Framer/MAC, serial backplanes, and more
      15. Wireless Communications - RF, base band, connectivity, transport and networking solutions for wireless equipment, addressing standards such as WCDMA, HSDPA, WiMAX and others.

### FPGA for Cloud Infrastructures

FPGA starts to appear recently in commercial cloud platforms such as Microsoft, Amazon, and others to respond to some cloud issues (improve power over a cluster of servers and the varying response times). The data centre deploys FPGA in their infrastructures with two main approaches, either FPGA tightly coupled to the Central Processing Unit (CPU) or FPGA as a standalone component.

The first approach considers FPGA as a coprocessor. FPGA and CPU are physically connected together, and the CPU is also connected to the network. In this case, FPGA becomes both an accelerator and a part of the data centre. However, the number of FPGA in the data centre is limited to the number of CPUs, and FPGA cannot be used as an independent computing resource. In this approach, Amazon has delivered the F1 instances type which integrates Xilinx Virtex UltraScale+. Amazon uses FPGA instances to accelerate some tasks ranging from analytics and machine learning to databases and network virtualization. The Amazon F1 instance achieves 10x better cost efficiency than Amazon’s CPU Elasticache. The CAPI solution from IBM is similar to that from Amazon; it deploys the Xeon processor with FPGA in the same package. Microsoft released the Catapult project in 2014 which deployed one Altera Stratix vFPGA per CPU. Microsoft connects FPGAs directly to the network but does not make them directly accessible and programmable by hardware designers. Catapult is difficult to be adapted to different applications. The target application accelerates the Bing web search engine with achieving an improvement of 95% in throughput improvement while consuming only 10% more power per CPU-FPGA server. These tightly coupled servers enable an acceleration of local applications to meet performance demands.[10]

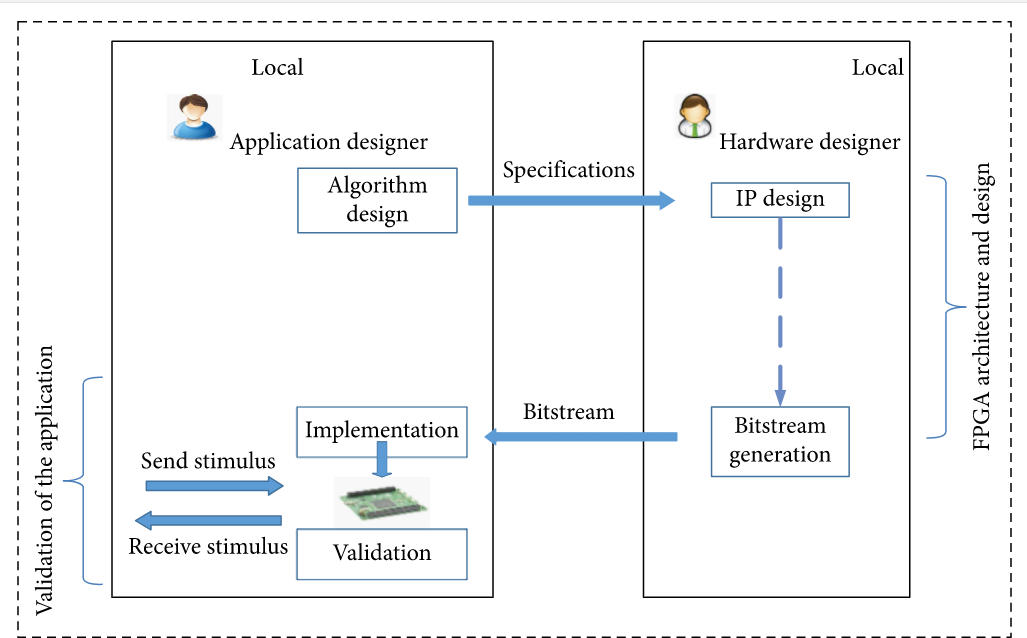
The second approach considers FPGA as a standalone disaggregated component independent from the CPU. FPGA is then directly connected to the network. This approach couples the network and the application processing in the same FPGA device. NARC is a standalone network-attached FPGA board designed for high-performance computing and network applications. The board consists of a Xilinx FPGA and an ARM processor. The ARM processor is used as a network interface to connect the FPGA to the network via an Ethernet interface. IBM

provides a new solution, which sets the FPGA free from the CPU to connect them directly to the data centre network. This system implements 16 platforms of FPGAs interconnected together via an Ethernet switch. This deployment shows that the applications can scale the number of FPGAs independently from the number of servers. Hence, it improves the latency and throughput, respectively, by 40x and 5x.

FPGAs are deployed with CPU on data centres to accelerate the execution time and minimize the data centre power consumption. In this case, cloud vendors are responsible for controlling the FPGA which hardware designers still do not have any access or control.

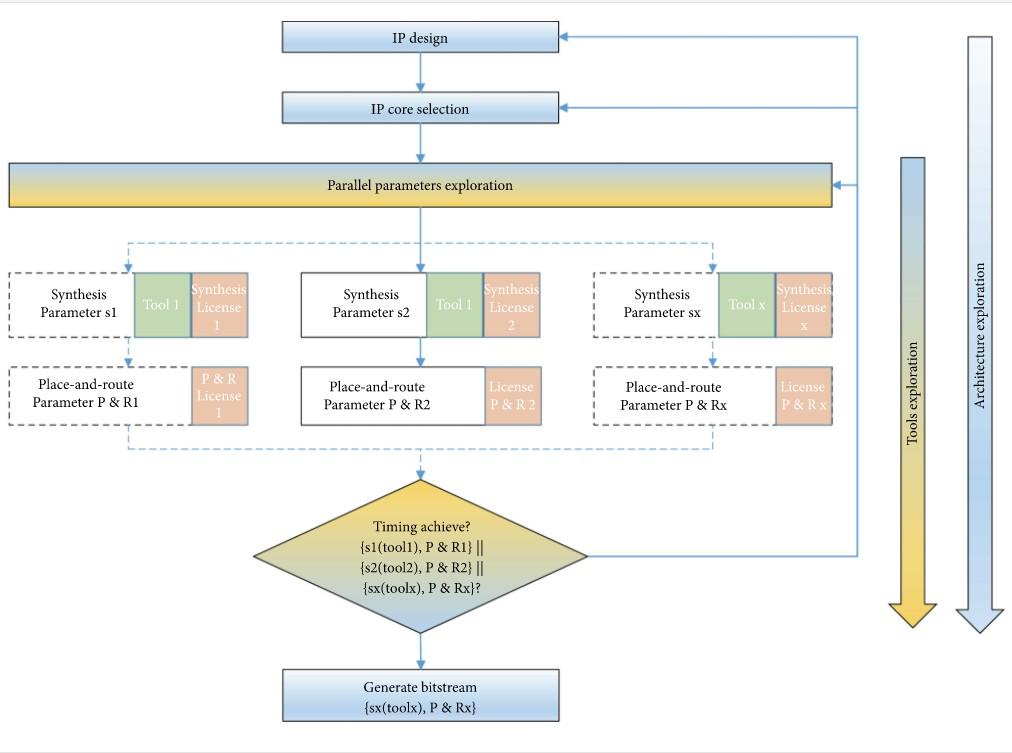
### Challenges for FPGA in Local

A process of FPGA implementation needs several competencies and interactions. In this article, we will describe it by reference to two types of designers. An *application designer* is in charge of describing the algorithm, writing specifications, and validating the implementation, whereas a hardware designer is in charge of the FPGA design flow from creating the IP design to performing FPGA implementation on board. For complex architectures, several application designers and hardware designers collaborate to design and validate the architecture on FPGA. The application designer first describes the algorithm according to the application required. Specifications of the algorithm are then given to the hardware designer. The hardware designer selects IPs or designs Intellectual Property (IP) if not existing. Functions of the algorithm are implemented by IPs. IPs are designed with an HDL and simulated, optimized, and tested with FPGA design flow. The bitstream, the programming information of the FPGA, is generated, and the end of the design flow and the application designer can download this bitstream on the FPGA to validate and execute the algorithm. The time required between the specifications and the generated bitstream is long especially if IPs must be designed or the implementation constraints are hard.



#### Fig 4.4 Interactions between application designers and hardware designers for FPGA design and implementation

**Challenges Related to FPGA Architecture and Design:** The hardware designer creates the FPGA architecture with the FPGA design flow and specific FPGA tools integrated in the flow. The objective is to design, simulate, and test the generated architecture and also to modify the design to meet the resource and timing constraints. Some steps and tools in the design flow lead to two main challenges in the FPGA architecture and design.



#### Fig 4.5 FPGA design flow with parallel parameters exploration.

The first challenge is linked to IP design (D1) and IP selection (D2). For IP design, the hardware designer describes the functions by means of IPs with hardware description languages. Such languages are completely different from languages used by application designers, and a strong hardware expertise is required. IP design is complex, requires IP competencies, and significantly increases the design cycle. Usually, several IP designers are required, and the continuous integration of designed IPs is done in the FPGA design flow, increasing the design cycle. For IP core selection, the hardware designer selects the appropriate IPs corresponding to the functions of the application designer. The IP selection can be made amongst in-house IPs and third-party IPs. For third party IPs, the hardware designer can buy different types of IPs (softcore, hard core, and firm core) from many specialized IP vendors. It is very common to get IPs from multiple vendors for a single design. The IP selection depends on different criteria such as the IP functionality, portability, the FPGA resources, and the frequency required. “Off-the-shelf” IP is not necessarily optimized for the application, and IP licenses come with various restrictions such as reuse, disclosure, and rights modifications which can possibly impair the design flexibility. The first challenge from IP design

and selection steps is called IPs store challenge (C1) where IPs in the architecture depend on many criteria like area, performance, and features, needed to be traded off against aspects like cost, license, risk, and time-to-market.

The second challenge mainly depends on tools and parameters. The FPGA design flow integrates several FPGA designs tools. Required tools in the design flow are synthesis, simulation, and place-and-route tools. FPGA design tools selection (D3) is required according to the availability of tools, the prices, and the functionalities of each of them. If several hardware designers are required, the FPGA design tools must be the same for all designers working on the same project. They should also have a strong expertise to use these tools. From these selected tools, another difficulty is the license software (D4). A same FPGA vendor offers several FPGA tools that can be integrated in the design flow. Buying an FPGA development kit enables the hardware designer to get specific licensed FPGA software with unlimited time and some specific functionalities. The hardware designer can also buy FPGA software without buying hardware platforms. He obtains the software with limited use in time. As an example, Intel provides to their hardware designers a set of FPGA development kit, the Stratix 10 GX, with one-year license software for the associated tools, the Quartus Prime Pro design software. For a design with multiple designers, a floating-point license with several seats is mandatory. For selected tools, a huge number of Tool parameters (D5) must be set. Indeed, FPGA design tools offer a huge set of parameters to guide implementation results such as resources, speed, power, and many other constraints. The hardware designer can modify these parameters to obtain different performance results. For example, he can obtain different resource consumption balance at the synthesis level, a better timing result at the timing closure level, or encrypted bitstream to preserve confidentiality of implementation at the generated bitstream level. The choice of parameters may significantly change according to the requirements, and these parameters must be selected according to the targeted design. Another difficulty is timing closure (D6). This metric refers to the process by which an FPGA is modified to meet timing requirements. Modern FPGAs with the support of millions of LUTs and thousands of DSPs and internal block RAMs require several iterations and lots of CPU time to find the right combination of settings. For instance, the learning-driven approach can be used to speed up convergence of timing closure. The main challenge from all these encountered difficulties is the FPGA design tools and parameters (C2).

### Cloud FPGA Services

The cloud FPGA is an infrastructure of FPGA devices or software design tools available in the cloud. In order to extract the benefits of “putting FPGA in cloud,” we propose to classify the cloud FPGA in three different levels of services:

* + - 1. FPGA Software tools as a service: From the SaaS cloud model, FPGA tools have been dematerialized since 2010. This model benefits from the massive computing of the cloud without worrying about tool incompatibilities and complicated setup steps. This model provides hardware designers an easy accessibility to the cloud without the complexity of infrastructure and software, but this model does not allow an access to a physical FPGA.
      2. FPGA Platforms as a service: From the PaaS cloud model, FPGA platforms have been dematerialized before 2010. There is no need to buy FPGA platforms. This model allows application designers to access one or several FPGA platforms. The designer can develop and implement their cloud applications on old or newer FPGA platforms.
      3. FPGA Resources as a service: From the IaaS cloud model, FPGA and its resources are virtualized since 2014. In this model, FPGA is divided into multiple independent virtual FPGA regions. These regions can be provisioned to multiple application designers in a multitenant environment with virtual access to the physical FPGA.

This classification may change with time according to different criteria due to the advent of new applications or changes in requirements. According to the service proposed for the cloud of FPGA, previous challenges can be solved.

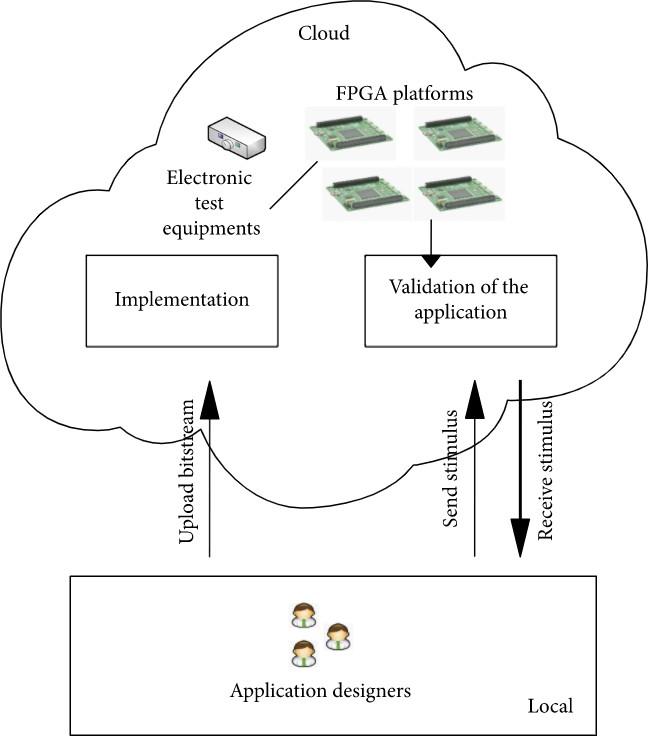
Cloud of FPGA is based on different levels:

1. At the service level, how to better support the idea of providing a new hybrid service with an easy selection of IP cores.
2. In the programming model, finding the best programming model for interconnecting IP blocks. It is interesting to determine what type of communications should be used to connect IP cores.
3. At the designer level, explaining the removal of designer’s interaction on both independent sides. The hardware designer is not in interaction with the application designer anymore and the design cycle only depends on the application designer.
4. At the management level, it is an important open question to identify a control system that can automatically supervise the platform and can interact with the environment.
5. Exploring how autonomously self-adaptive systems can be built that combine hardware designer capability with providing predesigned IP cores and the ability of application designers to execute their FPGA applications in cloud.

### FPGA Platforms as a Service

The issue associated with FPGA platforms is unavoidable as many academic institutions and companies have the problem of overcrowding. Several institutions are not properly funded; hence, they cannot provide the necessary number of FPGA platforms for every academic at the same time. In companies, some platforms are too big and too expensive to be deployed and provided for all designers. These issues (linked to D7 and D8) give birth to the idea of dematerializing FPGA platforms. Therefore, several institutions and companies have created their own remote FPGA labs. These labs offer a set of FPGA platforms with a remote visualization system, allowing application designers to quickly test and validate the new design.

In local, the hardware designers design their architecture with FPGA tools locally available. They generate the bitstream for the application designer. To access the cloud, the application designers upload the bitstream and validate the design by sending and receiving stimulus to the cloud. In most cases, each FPGA is connected to webcams and other similar types of materials to visualize and manage FPGA platforms. Application designers can control the FPGA platforms and experiments with the webcam in real time. In local, application designers receive output stimulus and evaluate the resources and they can check the timing performances.



#### Fig 4.6 FPGA Platform as a Service model.

In an Intel DE2 FPGA board including a Cyclone II device, the system targeted academics at introductory courses of digital design. ViciLogic is a remote academic platform which enables testing experiments over FPGA platforms. The ViciLogic has two versions. The most recent version (ViciLogic 2.0 prototype is integrated in the Xilinx Vivado tool) automates online and local SoC digital logic hardware prototyping. It modifies the HDL model to provide signal observability and integrates SoC resources (ARM, AXI interconnect, peripherals, and viciLogic IP). Vicilogic 2.0 has generated two bitstreams: one in Xilinx Zynq SoCs and one in Intel CycloneV SoC, eDiViDe is a similar platform which hosts multiple FPGAs from different universities. Each FPGA is connected to a camera and/or microphone for registering the behaviour of the platform. Another recent academic remote lab focuses its works in image processing such as the lane detection in road scenes applications. The application designers upload the bitstream generated via the internet to the

remote lab server. The server programs the FPGA with the bitstream and outputs the processed image. The remote lab allows application designers to compare resources used and power consumption on different types of FPGA platforms. Machidon et al. use a single FPGA platform connected to the internet used for measuring functional parameters. The hardware designer processes the design flow steps and generates the bitstream. Then, the application designer uploads the bitstream on the cloud and executes and validates the design. Several companies like Synopsys and Cadence provide large FPGA platforms dedicated for several complex application implementations. Zebu platform, HAPS, and proFPGA are major existing hardware emulator platforms. As a result, the application designer uses one of these platforms remotely by using an Ethernet controller to benefit from highest performances and a low-cost solution. Amazon provides an FPGA virtual machine image for its cloud FPGA instances, where users can easily develop and deploy FPGA acceleration applications. Recently, Microsoft company presented AccelNet, which is an FPGA-based platform for host SDN processing supported by the software and the hardware infrastructure of the previous catapult project. Microsoft Azure also offers an FPGA-based platform to enable application designers to deploy machine learning applications. A key challenge is the multitenancy to efficiently share the FPGA while enforcing strict data and performance isolation between tenants. Alibaba cloud officially launched three generations of large-scale FPGA instances, respectively, Ali F1, Ali F2, and Ali F3 based on Intel and Xilinx FPGAs to achieve strong isolation between IP acceleration and the deployment environment.

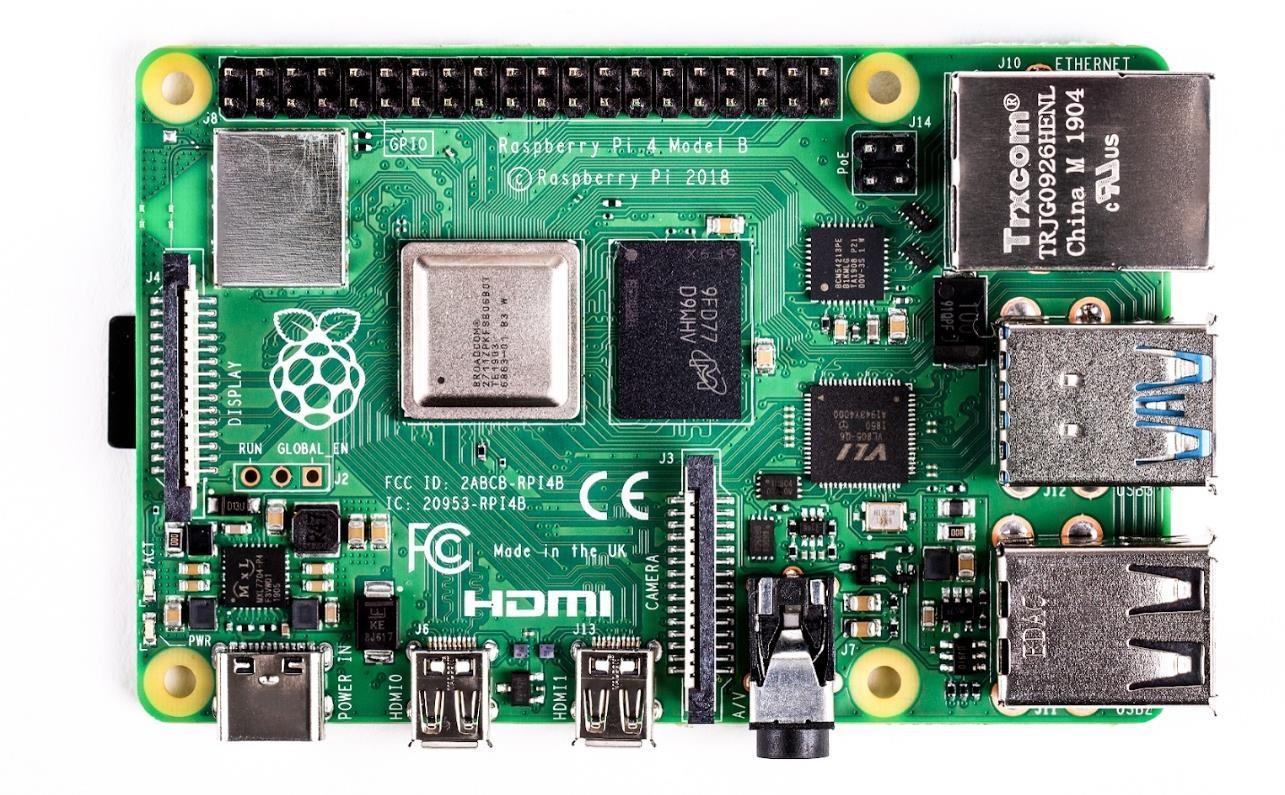
Many remote labs exist and are mainly dedicated to academic institutions and industries in order to mutualize FPGA platforms and devices. Often a remote FPGA lab provides a set of identical FPGA platforms with a limited number. The applications are executed in the cloud at the same time but for a limited time. While platforms are still valuable for small design to academic researches, industrial teams can use a powerful platform for debugging and implementing complex designs. These powerful platforms include a huge number of FPGA devices; therefore, multi- application designers can implement their designs without worrying about the choice of FPGA device and platform. In spite of these features, a designer must have hardware expertise to develop designs and also to manage the FPGA tools locally.

## RASPBERRY PI

Raspberry Pi is the name of a series of single-board computers made by the Raspberry Pi Foundation, a UK charity that aims to educate people in computing and create easier access to computing education.

The Raspberry Pi launched in 2012, and there have been several iterations and variations released since then. The original Pi had a single-core 700MHz CPU and just 256MB RAM, and the latest model has a quad-core 1.4GHz CPU with 1GB RAM. The main price point for Raspberry Pi has always been $35 and all models have been $35 or less, including the Pi Zero, which costs just $5.

All over the world, people use Raspberry Pis to learn programming skills, build hardware projects, do home automation, and even use them in industrial applications. The Raspberry Pi is a very cheap computer that runs Linux, but it also provides a set of GPIO (general purpose input/output) pins that allow user to control electronic components for physical computing and explore the Internet of Things (IoT).[17]



### Fig 4.7 Raspberry Pi 4

The Raspberry Pi 4 Model B is the latest version of the low-cost Raspberry Pi computer. In its cheapest form it doesn't have a case, and is simply a credit-card sized electronic board -- of the type user might find inside a PC or laptop, but much smaller.

|  |  |  |
| --- | --- | --- |
| **Pin Group** | **Pin name** | **Description** |
| Power source | 5V, 3.3V, GND and Vin | 5V – power output 3.3V – power output |
| Communication Interface | UART, Interface (rxd , txd  )[(GPIO15, GPIO14)] | UART(Universal Asynchronous Receiver Transmitter) used for interfacing sensor and other devices |
| SPI interface (MOSI , MISO , CLK, CE  [SPIO11, GPIO8)] | SPI(serial peripheral interface)(rxt,dxt) | - |
| TWI Interface(SDA, SCL) x 2 [(GPIO2, GPIO3)]  [(ID\_SD,ID\_SC)] | TWI (Two Wire Interface) Interface can be used to connect peripherals. | - |
| INPUT OUTPUT PINS | 26 I/O | Although these some pins have multiple functionsthey can be considered as I/O pins. |
| PWM | Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19 | These 4 channels can provide PWM (Pulse Width Modulation) outputs.  \*Software PWM available on all pins |
| EXTERNAL INTERRUPTS | All I/O | In the board all I/O pins can be used as Interrupts. |

#### Table 4.1 Raspberry pi pin configuration

* + 1. **WORKING OF R-PI**

The Raspberry Pi 4 can do a surprising amount. Amateur tech enthusiasts use Pi boards as media centres, file servers, retro games consoles, routers, and network-level ad-blockers, for starters. However, that is just a taste of what's possible. There are hundreds of projects out there, where people have used the Pi to build tablets, laptops, phones, robots, smart mirrors.

With the Pi 4 being faster, able to decode 4K video, benefiting from faster storage via USB 3.0, and faster network connections via true Gigabit Ethernet, the door is open to many new uses.

It's also the first Pi that supports two screens at one -- up to dual 4K at 30 displays -- a boon for creatives who want more desktop space.

#### SD CARD

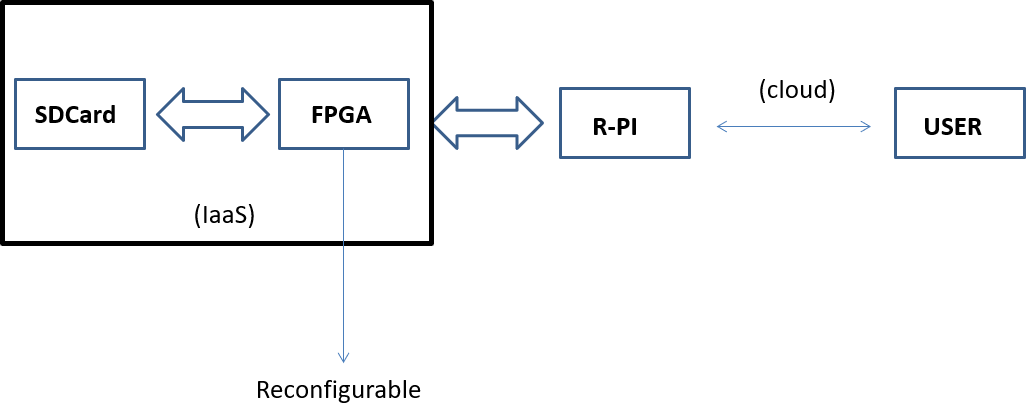


**Fig 4.8 SanDisk Extreme 64GB**

A type of removable flash memory card designed specifically for mobile phones. Like any flash memory card, it can be used to store various types of files, including photos, videos, music, or software. microSD is one of the smallest memory card formats available; a microSD card is about the size of a fingernail. It was designed to be smaller than competing formats, to allow phones using the format to be smaller overall. SanDisk claims this microSD card can read up to 95MB/s and write 90MB/s, and in tests it came pretty close to those speeds. This makes it a very fast card, and with smaller files it's even faster, which means this is a great card for transfer speeds.

# CHAPTER - 5 MDD VIEWPOINTS

## SYSTEM MODELLING



### Fig 5.1 Architectural modelling of the system

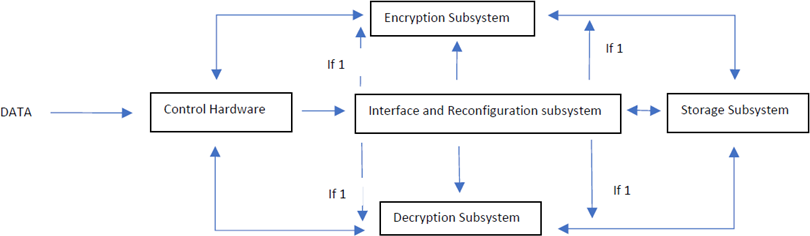
The architecture includes the major physical components of the system: Field Programmable Gate Array, Raspberry pi, Secure Digital card and the data from the user that needs to be processed and stored. The data from the user is sent via a secure internet connection and is received to the Raspberry pi which is connected to the FPGA. As the data is received, R-pi sends the data to the FPGA for encryption after acknowledgment and FPGA performs the required encryption technique and the encrypted data is stored in the SD card which is used for cloud simulation in the prototype.

When the user requests the stored data, R-pi intimates the FPGA and required decryption techniques are performed and the original data is retrieved and sent to R-pi from SD card and R-pi in turns proceeds with sending the data to the requesting user.

## DECOMPOSING THE SYSTEM – IDENTIFYING THE MAJOR SUBSYSTEMS

The entire system can be divided into five major subsystems based on their operation at any point. They are:

* + 1. Control hardware
    2. Interface and reconfiguration subsystem
    3. Encryption subsystem
    4. Decryption subsystem
    5. Storage subsystem



#### Fig 5.2 Architecture Modelling of Subsystems

**Control hardware:** The Raspberry pi serves as the major control hardware and it interacts with three subsystems namely Interface and reconfiguration subsystem, Encryption subsystem and Decryption subsystem in half duplex communication. The control hardware receives the data and intimates the Interface and reconfiguration subsystem and based on the signal from IRS, the data is sent to the encryption subsystem. Similarly, when the user requests back the data, another intimation is given to IRS by the control hardware and the data is retrieved from the decryption subsystem by the control hardware and sent back to the user.

**Interface and reconfiguration subsystem:** A part of FPGA acts as the interface and reconfiguration subsystem here. This is the only subsystem which interacts with all the other subsystems. When the intimation is received from the control hardware to send or receive the data, necessary control signals are given to the required subsystem to process the data, store the data and in retrieval of the same.

**Encryption subsystem:** When control signal is received from IRS, encryption subsystem receives the data from the control hardware, encrypts it and sends it to the cloud storage in a pipelined fashion.

**Decryption subsystem:** When control signal is received from IRS, decryption subsystem receives the encrypted data from the storage subsystem, decrypts it and sends it to the control hardware which in turn sends it to the requesting user.

**Storage subsystem:** The encrypted data is stored here which is approx. 33% larger than the plain text based on the key size. Only when the control signal from IRS is sent to the decryption subsystem, this encrypted data can be retrieved from here. As soon as the encrypted data is decrypted and sent, the storage space becomes empty.

## MATHEMATICAL MODELING AND GOVERNING EQUATIONS OF THE SYSTEM

The major mathematical modelling is with the math behind the various cryptographic algorithms used in different reconfiguration of the FPGA. An overview of the cryptographic mathematical model is explained below:

We want to give a formal definition of the following two items:

* + 1. An encryption function transforms arbitrary character strings into other character strings. (Where the strings are from a given alphabet.)
    2. A cipher is a parametrized family of encryption functions. The parameter is called the key. It determines the choice of a function from the family.

The purpose of this construct is that nobody can invert the encryption function except people who know the key. That is, an encrypted message (or a text, a file . . . ) is kept secret from third parties. They can see that there is a message, but they cannot read the contents of the message because they don’t have the key and therefore don’t know which of the functions from the family to invert.[12]

### Alphabets and Texts

Let Σ be a finite set, and call it alphabet. Call its elements letters (or symbols, or characters). Examples. Here are some alphabets of cryptographic relevance:

* + - 1. {A, B, . . . , Z}, the standard 26 letter alphabet of classical cryptography.
      2. The 95-character alphabet of printable ASCII characters from “blank” to “tilde”, including punctuation marks, numbers, lowercase, and uppercase letters.
      3. {0, 1} = F2, the alphabet of bits, or the field of two elements. The earliest appearance (after Bauer) is Bacon 1605.
      4. F52, the alphabet used for telegraphy code since Baudot. It has 32 different symbols and also goes back to Bacon (after Bauer).
      5. F82, the alphabet of bytes (correctly: octets, because in early computers bytes did not necessarily consist of exactly 8 bits). The earliest appearance seems to be at IBM around 1964.
      6. More generally Fl2, the alphabet of l-bit blocks. Often l = 64 (for example in DES or IDEA), or l = 128 (for example in AES). See Part II. Often the alphabet Σ is equipped with a group structure, for example: K. Pommerening, Monoalphabetic Substitutions 3
      7. Zn, the cyclic group of order n = #Σ. Often we interpret the calculations in this group as arithmetic mod n, as in elementary Number Theory, and denote Zn by Z/nZ, the residue class ring of integers mod n.
      8. F2 with the field addition +, as Boolean operator often denoted by XOR or ⊕. (Algebraists like to reserve the symbol ⊕ for direct sums. For this reason, we’ll rarely use it in the Boolean context.)
      9. Fl2 as l-dimensional vector space over F2 with vector addition, denoted by +, XOR, or ⊕. For an alphabet Σ we denote by Σ∗ the set of all finite sequences from Σ. These sequences are called texts (over Σ). A subset M ⊆ Σ∗ is called a language or plaintext space, and the texts in M are called meaningful texts or plaintexts. Note that the extreme case M = Σ∗ is not excluded.

### Ciphers

Let K be a set (finite or infinite), and call its elements keys.

#### Definition

An encryption function over Σ is an injective map f : Σ∗ −→ Σ∗.

A cipher (also called encryption system or cryptosystem) over Σ with key space K is a family F = (fk)k∈K of encryption functions over Σ. Let F be a cipher over Σ, and F˜ = {fk|k ∈ K} ⊆ Map(Σ∗, Σ∗) be the corresponding set of different encryption functions. Then log2(#K) is called the key length, and d(F) = log2(#F˜), the effective key length of the cipher F.

### AES

The Mathematics needed to understand AES:

Fields:

* + - 1. Real Numbers and Rational numbers are fields
      2. Set of values such that addition, subtraction, multiplication, division can be applied to values in it
      3. Other fields exist, and can be defined

Finite Fields: A field, with a finite number of elements (unlike Real #s) Modular Arithmetic

1. 9 Ξ 14 Ξ 24 Ξ 4 (mod 5)

2. Works on integers, but also works on other groups and fields The Cipher for AES involves:

1. Rijndael's Finite Field
2. a 'characteristic 2 finite field with 8 terms' (strings of 8 bits)
3. The Galois field GF(2^8)

Addition done with XOR operator: Uses the reducing polynomial x8 + x4 + x3 + x + 1 for multiplication (an irreducible polynomial in GF(2^8))

For a Higher-Level Finite Field:

1. Polynomials with Coefficients in GF(2^8) 2. a(x) = a3x3 + a2x2 + a1x + a0

3. Each of ai are bytes, elements of GF(2^8) Addition:

a(x) + b(x) = (a3 ⊕b3 )x3 + (a2 ⊕b2)x2 + (a1 ⊕b1)x + (a0⊕b0)

Multiplication:

1. Same as multiplying two polynomials
2. We reduce to degree 4 by using mod x4+1
3. For fixed polynomial, multiplication can turn into Matrix Vector multiplication

## DESIGN VARIABLES

The security of cloud computing is defined by up to 4 design variables, thereby dramatically increasing the complexity of the model. Design variables are those components that change the whole dilemma of the system. In this case there are three main such components.

**Timing**: Reconfiguration factor and timing required to perform encryption/decryption are directly proportional. More security in the system requires more encryption and since time factor and encryption are directly proportional more the encryption more time it takes to processes.[8]

**I/O count**: Parallel computing refers to the process of breaking down larger problems into smaller, independent, often similar parts that can be executed simultaneously by multiple processors communicating via shared memory, the results of which are combined upon completion as part of an overall algorithm. High I/O count in FPGA will result in parallel computing. Therefore, the number of I/O count in FPGA plays a major factor in this system.[9]

**Key size**: The key size affects the storage capacity. The encrypted file is base64 encoded which would account for 33.3% file increase. Inserting a new line every 64 characters to make it easier to read (as is done by ASCII Armor in opens’, GPG, PGP) will increase the size by 65/64.Combining these two effects results in the new file being (4/3) \*(65/64) = 135.4% of the size of the original or an increase in file size of 35.4%. If the file size of a data is more so that the sd card can’t store the data after encryption, Then It will be a discrepancy. The size of the encrypted data is directly proportional to the size of the key. For a secure system the encryption should be as good and thereby key size is one of the major factors in this model.

## CONSTRAINTS

Theory of Constraints (TOC) is a new concept. Every project has some constraints. Effective design constraint requires design analysis and restraint to develop and maintain the correct constraint balance. Over-constraining a design will cause the tools to work harder to resolve conflicting or unreasonable requirements with limited resources. Design over-constraint can occur in several different ways. Some of the most common include simply assigning too many constraints, constraining noncritical portions of the design, and setting constraints beyond the required level of performance.

Over-constraining a design can result in a significant increase in the time required to place, route and analyse a design. The result is a longer design implementation time. Since the design implementation phase potentially occurs many times during a design cycle this can have a significant impact on design efficiency. A more serious design over-constraint consequence occurs when the place-and-route process can no longer successfully implement the design within the specified FPGA architecture. This may force an upgrade to a larger or faster speed-grade FPGA component if the over-constraint conditions are not adjusted.

**Timing Constraints**: Timing constraints may be used to influence and guide the placement of design elements and signal routes between placed elements in order to meet design performance requirements.

The two general types of timing constraints are global and path-specific. Global timing constraints cover all paths within the logic design. Path-specific constraints cover specific paths.

* + 1. Identify and constrain system clocks. The timing constraint process should start with the specification of the global timing constraints for all identified system clocks.
    2. Identify and create signal path groups. The two primary types of path groups are global and specific. A global group typically includes a group of paths between registers, input paths, and output paths. Ideally these paths should be within the same clock domain. Specific paths are mostly static or combinatorial paths, paths between clock domains, or multicycle paths. Multicycle paths are defined as paths between logic elements that have a timing requirement that is a multiple of the clock period for the logic elements. For example, if a series of logic functions require more than a single clock cycle to complete, the data will be correct at the circuit output.
    3. Assign global constraints. The general rule of thumb when assigning constraints is to use global constraints for primary coverage of a majority of the design paths. Apply global

period constraints to the design before the HDL synthesis phase. With access to timing constraints, synthesis tools may attempt to optimize the synthesized design to meet the specified timing requirements.

**Key Constraints**: As additional data is embedded along with the plain text for encryption, the storage space required is relatively larger than the storage space for the plaintext. The encrypted file is base64 encoded which would account for 33.3% file increase and will increase the size by 65/64. [7]

**Pin Constraints**: This is a common question for designers to ask, since the FPGA tools are trusted to place and route the design. However, there are several factors that influence software-controlled resource location assignment. One of the primary FPGA placement directives is to spread functionality out to avoid routing congestion. With no clear guidance to the contrary, the tools will typically work to spread functionality out across the available resources. As an example, FPGA tools can have difficulty identifying the pins that make up a signal bus and can also have difficulty identifying the control signals associated with the bus. Without knowledge that the signals form a group, the tools do not seek to co-locate the signals even though they may benefit from closer placement. While it may be possible to increase the global constraints of the design so that the bus signals and related control signals will be located as a group, the design team then runs the risk of over-constraining the design. This can significantly increase the place-and-route time for the FPGA software.

The process of I/O assignment is more involved than simply assigning signals to available package pins. Assigning board-level signals to FPGA I/O can have a large impact on system performance. In an ideal world, the critical FPGA functionality would have already been captured, compiled and simulated multiple times before the pin assignment step, allowing the design team to determine an optimized pin assignment. However, in a typical rapid system development, device pins are assigned early in the design cycle. The early assignment may be necessary to support early PCB layout. It is possible for the PCB board to have already been routed and in the process of being built before a significant percentage of the FPGA functional design has been captured. This “pin- locking” may be required to meet aggressive design schedules and allow the FPGA development to occur in parallel with the board build effort. This has the effect of maximizing schedule progress, while also increasing risk. It is important to note that pin assignment is not critical for all designs, or all the pins in a design. Designs with significant I/O margins or slow operational speeds may not

require careful pin assignment. However, pin assignment may become a critical factor if the design margin is limited by any of the following FPGA design factors:

* + - 1. I/O pin availability
      2. FPGA fabric-level logic resources

#### Design Constraints and Optimization

1. On-chip routing resources
2. Required logic speed versus maximum FPGA speed
3. Required logic speed versus layers of logic required to implement the design

Pin assignment can also become critical at the board level when signals require special routing considerations such as short signal trace length, matched line length, or controlled impedance. These requirements might be a result of signal loading or speed requirements or EMI requirements. Most designs fall into a crossover group where pin assignment is not quite critical but also not an insignificant factor in design performance. Almost any design can benefit from a well implemented pin assignment. It is possible to affect and improve design performance through considered pin assignment. The design factors that may influence pin assignment include:

1. The size of the device
2. The device package required
3. The speed grade of the device
4. The maximum speed that the FPGA can run

## COSTING AND FINANCIAL MODEL

No project starts without a budget. Project success is decided by how well the project cost has been handled in the project. Many times, it happens that the project may not be completed within the project cost. It means that when compared the Project Cost Vs Project Profit, Project Cost might have exceeded and it is of course considered as a project failure. Hence, it’s very important to come up with the correct cost estimation needed for the project.

To come up with accurate cost estimation, it’s required to understand the types of project costs involved in the project.

### Types of Project Costs:

There are majorly 4 types of project costs incurred in any project. They are

* + - 1. Fixed Cost
      2. Variable Cost
      3. Direct Cost
      4. Indirect Cost

**Fixed Cost**: Any Cost which is fixed throughout the project life cycle and would not change by quantity, time or any other project factors called for a fixed cost. Fixed costs include major components like FPGA and Raspberry Pi.

**Variable Cost**: The Variable cost is a cost which varies or changes in proportion to product or service that the project produces. Storage infrastructure of the system may vary based on demand and so expansion cost contributes to the variable cost.

**Direct Cost**: Costs which are directly visible and accountable to produce the project output are called direct costs. Human resources involved in large scale implementation of the proposed solution contributes to the direct cost.

**Indirect Cost**: Costs which do not directly contribute or specific to the output of the project are called indirect costs. It may be either variable or fixed.

|  |  |
| --- | --- |
| **Apparatus** | **Cost** |
| FPGA – Edge spartan 6 development board | Rs. 7500 |
| Raspberry pi 4 – model B | Rs. 3000 |
| Sandisk memory card 64GB | Rs. 1100 |
| **Total** | Rs. 11600 |

**Table 5.1 Cost model for the prototype**

# CHAPTER 6 CONCLUSION

In the modern-day technology cloud plays a vast and major role. All data from personal to workspace is stored in the cloud. In 2018, it found, there were a total of 11.8 billion records exposed with a total cost of $1.76 trillion. By 2019, that number rose to 21.2 billion exposed records, and the cost rose to $3.18 trillion. Nearly 70% of organizations that dealt with breaches were founded before 2010 while about 7% were companies inaugurated after 2015.When such is the case, a secure environment for Cloud emerges as the basic concern.

By adding an additional layer of the hardware that is FPGA and improving the encryption and decryption standards of the system by using advanced encryption techniques like the AES, DES and RSA cloud computing will become more secure. Since this model also uses reconfigurable computing major problems like data breaches , data mis-configuration , data insecurity is solved thereby making cloud computing environment much more safe and a secure environment.

## LIMITATIONS

* + 1. In an increasingly software dependent world, establishing a hardware layer into the system is a risky process.
    2. Delay in retrieval of data is significant unlike the existing resources.
    3. Large scale implementation requires a lot of monetary and human resources.

## FUTURE SCOPE

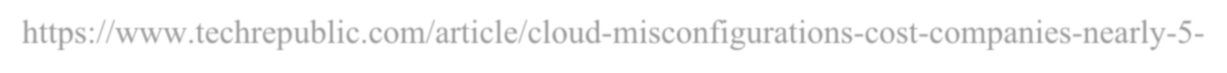
There are many other related works and applications where the proposed could provide extra performance and additional layers of security. FaaS can offer security through deep packet inspection if a network interface is directly accessible, but FaaS offers much more as all data outside of the FPGA is encrypted. Even though data processing in a FPGA is not guaranteed to be fully secure, it improves upon existing cloud techniques by making both insider and outsider attacks more difficult. Combined with a weak fully homomorphic scheme, or the distributed approach of MPC, gives even greater levels of data privacy. With secure FPGA as a service, users would gain greater privacy and security when using third-party cloud services, enabling them to have more control over their data.

## REFERENCES

1. <https://www.cloudflare.com/learning/cloud/what-is-the-cloud/>
2. <https://opencirrus.org/4-types-cloud-computing/>
3. <https://searchcloudcomputing.techtarget.com/definition/Software-as-a-Service>
4. <https://azure.microsoft.com/en-in/overview/what-is-iaas/>
5. <https://azure.microsoft.com/en-in/overview/what-is-paas/>
6. <https://www.cloudflare.com/learning/serverless/glossary/function-as-a-service-faas/>
7. <https://www.geeksforgeeks.org/issues-in-cloud-computing/>
8. <https://asq.org/quality-resources/design-of-experiments>
9. <https://www.omnisci.com/technical-glossary/parallel-computing>
10. <https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html>
11. <https://www.hindawi.com/journals/ijrc/2019/8085461/>
12. <https://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/1_Monoalph/MathModel.pdf>
13. <https://booksite.elsevier.com/samplechapters/9780750678667/Sample_Chapter.pdf>
14. <https://www.geeksforgeeks.org/cryptography-and-its-types/>
15. <https://securityboulevard.com/2020/05/types-of-encryption-5-encryption-algorithms-how->

to-choose-the-right-one/

1. <https://www.educba.com/what-is-decryption/>
2. <https://opensource.com/resources/raspberrypi>



[18] https://www.techrepublic.com/article/cloud-misconfigurations-cost-companies-nearly-5-



trillion/#:~:text=The%20company's%20researchers%20cited%20data,and%20processes%20to%



20cloud%20platforms