Applicability of Homomorphic Encryption and CryptDB in Social and Business Applications: Securing Data Stored on the Third Party Servers while Processing through Applications

Kurra Mallaiah
Osmania University, Hyderabad,India

S. Ramachandram
Osmania University, Hyderabad,India

ABSTRACT

Confidentiality in third party services like cloud computing has become a major concern. IT industry and government organizations are very serious about security factor in cloud computing, because its usage has reached all the way from a common man having a mobile phone to large scale business enterprises. In this paper, we present security threats in social and business applications accessing the data stored in cloud computing scenario. Also, we critically discuss homomorphic encryption and CryptDB schemes which are applicable to protect data from malicious third party service environments (cloud computing) and also from insiders for these applications. We also present empirical results of partial homomorpic encryption algorithms over one lakh 10-digit numbers, using Linux virtual machine on VirtualBox, VMPlayer and KVM. The result for four algorithms (namely Paillier, ElGamal, RSA and Benaloh) as performed on the above four different platforms are computed to show their respective overhead values as compared to plain data operations. In case of Paillier Algorithm the overhead is 17, 15, 22 and 12 times for addition operation and 278, 399,518 and 346 times for multiplication operation respectively. Similarly, in case of Elgamal algorithm 1.72, 1.6, 11.7 and 8.9 times for multiplication operation; in case of RSA algorithm 1.79, 1.5, 3.48 and 1.5 times for multiplication operation and in case of Benaloh algorithm is 5.6, 5.36, 5.48 and 3.5 times for addition operation respectively. These performances clearly indicate that these algorithms are quite feasible enough to be used in context of social and business applications by third party service providers

KEYWORDS

Homomorphic encryption, CryptDB, Cloud Computing security, Social and Business applications

1. INTRODUCTION

Most of the people think it would be nice if somebody else can do their work so that by outsourcing some of the work they can concentrate more on critical work. Even people are thinking of ready-made solutions and they should be made available within short time. This kind of scenario is very common in IT industry today, where people are looking for readily available resources like computing, storage and applications. Cloud computing is a potential solution for this kind of requirements. It provides various services like Software as a Service, Platform as a Service, Security as a Service, Infrastructure as a Service and even one can think of everything as Service (XaaS). Despite of many advantages and uses with cloud computing, there is still a doubt in the mind of a potential users and industry about the safety of their application and confidentiality of data hosted on the cloud computing? Most critical problem in cloud computing is of confidentiality, it is because of outsourcing the applications and data onto the service provider's premise, where the customers are losing the physical control of their applications and data. The Homeland Security Newswire listed that between 2009 and 2011, 8 million medical records were leaked [1]. In another example, a group of hackers infiltrated into the Sony play station network and were able to access 77 million user profiles [2]. In 2012 a social networking website LinkedIn member's passwords have been compromised and more than six million passwords leaked onto the internet [32]. According to the Verizon Data Breach Investigation Report (DBIR), internal agents were involved in almost half of data breaches [56].To give some more examples, attackers may use the vulnerability in the service provider's server software to interruption in [59], there is possibility that a malicious administrator may glance at the data on the service provider's server [60,61], or the service provider or server agent may be compelled to unveile data by law [36].Therefore, major confidentiality problem in cloud environment is from administrators, insiders, hackers and vulnerability in the server side software. There is a probability that in a different ways the user confidential data in service provider's environment may be leaked or misused. So, cloud users are constantly subjected to insecure use of such cloud computing services. Therefore to build the confidence in cloud users, it is mandatory to protect their applications and data from unauthorized users, malicious or corrupt administrators and from any other security breaches. Data at any point in time i.e. data in transition, data at rest and data in processing should be encrypted and decryption should only be done by the user if ever he wishes to. The service providers should perform computation (if any) on the encrypted data itself. This will provide complete confidentiality to the user data. The traditional encryption mechanism provides the confidentiality of data by encrypting it but, does not support any computations on encrypted data and for any required computations data needs to be decrypted. Therefore, it is required to focus on homomorphic techniques of encryption, where it can support computations on encrypted data itself. In this scenario, the application will send an encrypted query to the service provider which will then be executed on the encrypted data directly. The results will be sent by the cloud in encrypted form to the user where encrypted results will be decrypted. This mechanism ensures complete confidentiality, because user data never leaves encrypted form in the service provider's environment and the crypto keys are also not shared with the service provider for any decryption. In this paper, firstly, analyzed general cloud computing security scenarios; social and business applications hosted in third party services and involved security threats (corrupt administrators, malicious

programs, spurious insiders, side channel attacks, hardware Trojan horses, Zero-Day attacks etc.). Secondly, we have critically discussed state of the art partial encryption, fully homomorphic encryption and CryptDB schemes and their respective applicability in third party service environment with respect to above explained threats. We also present empirical results of four partial homomorpic encryption algorithms over

one lakh 10-digit numbers, using Linux virtual machine on VirtualBox, VMPlayer and QEMU/KVM. The results show that partial available homomorphic encryptions techniques are practical and these algorithms can be used in social and business applications.

2. GENERAL CLOUD COMPUTING SECURITY SCENARIO

In a general cloud computing security scenario, customers use online database services for hosting their data in cloud computing. Let us consider i) 'X' is a cloud service provider and provides databases online. ii) 'X' is also a security aware merchant, so he maintains all user data in encrypted form, both when 'X' sends a reply for user query and when saves information on to the disk. These days, this kind of security approach is quite evident in majority of cloud service providers. Now, whenever a cloud user searches for some data in the database hosted on the cloud computing environment process will be as following. Firstly, the user encrypts the search term and uploads it to service provider. Secondly, service provider decrypts the search term so that he knows what to search for. Thirdly, the service provider decrypts the user data stored in his database and performs the search operation using the decrypted data. Lastly, the service provider encrypts the search results, if any, and returns them to the user. Subsequently, all the search terms and the decrypted data is assumed to be cleaned once the search is complete. Moreover, it is also assumed that the service provider doesn't take any kind of undue advantage through decrypted user data. But, obvious questions still arises under following situations. In case some malicious code or program is present in the service provider's environment, it could be used for any organizational or government advantage or could be used by competitors/rivals. Thereby, any intentional or unintentional attack is very possible. These are few threat scenarios which mainly arise due to decryption of the user data inside third party environments.

3. SOCIAL AND BUSINESS APPLICATIONS AND THEIR THREAT SCENARIO IN CLOUD COMPUTING ENVIRONMENT

Uses of cloud computing for hosting social, business applications are increasing day by day. In this section, discussed usage of cloud computing in social and business applications and associated threat scenarios.

3.1 Financial institutes

Now a day, use of online banking system has been increased drastically. Due to supportive government regulations, directly linked government schemes and transparency in the payment system, most of the people are motivated to posses bank accounts. Moreover, in this age every common man opens his bank accounts to safeguard his assets. There is a fair competition among bankers in terms of providing services to customers is quite obvious. All these reasons lead to enormous computing and storage requirements in the current banking system. Here, cloud computing has offered promising prospects in this direction. Financial incapability of banks for procuring and maintaining private infrastructure and numerous other advantages with cloud computing is leading the contemporary bankers to switch to cloud computing paradigm for their businesses. But, their grave concern is IT security associated with cloud computing services. The security problem is not only with outsourcing but also from internal corrupted or spurious employees.

3.1.1 Threat Scenario

Designated employees in any bank are able to access confidential information related to customer accounts. The catch here is that any spurious bank employee can misuse the confidential information available at his dispense for his private gains. Similarly while using any third party services, service provider may misuse the customers vital information while processing a database query or otherwise accessing information during some computation.

3.2 University Examinations System

Now a days, Examination Cell in universities use web based solutions to carry out their administrative activities such as online receiving of applications, distribution of hall tickets, declaration of results etc. Therefore, examination cell has to store and process lot of information pertaining to their students. Here, the examination cell can store the information and carry out information computation by making use of cloud services.

3.2.1 Threat Scenario

The huge banks of information hosted by the examination cell are kept in plain or if not they are decrypted for required computations. Given the fact the students usually remain inclined towards maliciously accessing this information; a potential threat is clearly evident. Another threat is that malicious administrator may modify or misuse the information of students for unscrupulous gains.

3.3 Mobile Communication

Mobile Cloud Computing Forum describes Mobile cloud computing as [51]: "An infrastructure where both the data storage and the data processing happen outside of the mobile device. Mobile cloud applications move the computing power and data storage away from mobile phones and into the cloud, bringing applications and mobile computing to not just smart phone users but a much broader range of mobile subscribers". The applications and data which are hosted on cloud computing can be accessed through mobile phone through mobile browsers. In the following section, some mobile cloud computing applications and their security threats are discussed.

3.3.1 Mobile Commerce

As per current trend, people are looking for the technology which solves most of their requirements using mobile phones. They are trying to perform most of the jobs online; such as paying bills and shopping etc. One of the successful mobile business model is mobile commerce (m-commerce).This is basically doing business using mobile phones. The m-commerce applications are being used for bill payments, mobile banking, shopping, ticket booking etc. In fact, vendors are starting to use more and more m-commerce applications in addition to those in e-commerce. There are some critical challenges being faced by m-commerce applications such as high computational requirement, low network bandwidth, storage requirements, and most importantly security. To achieve computational and storage requirements in m-commerce applications one can utilize cloud computing services, where m-commerce applications are hosted on cloud and users will access these applications through their mobiles phones (e.g. using web browsers, vendors developed

applications). The customers can avail m-commerce applications on mobility with great comfort by taking help of cloud services.

3.3.2 Threat Scenario

The paradigm of m-commerce handles confidential information of customers. Any misuse of this information might result into catastrophic outcomes for customers and merchants. Mostly, m-commerce applications are hosted on cloud and the cloud server administrator will be managing customer's information during any troubleshooting in the system. Most of the m-commerce computations are done on the cloud servers using customer information in plain. Thus any malicious administrators or even at time malicious insiders can misuse the information for his unscrupulous gains.

3.3.3 Mobile health Care

Consider a situation where a patient is out of station and he needs an emergency medical service, naturally, he approaches to a nearest hospital for medical attendance. In that critical situation for a common man explaining the nature and status of his health condition is a very complicated task and sometimes patient may not be in a position to explain his previous medical conditions. It would be very nice that all of his medical records are stored in online database and he can able to access them as and when it is required online wherever he is. Cloud computing offers an emerging solution for this kind of requirements. Where hospitals, individual or any third party can store patient related data on cloud computing. This information can be accessed very well using mobile phones as and when and where it is required. Mobile health care, also called m-health care, make available mobile users who can access their health records online as and when they required with minimum effort and cost. There are lot of competition in the health industry and therefore hospitals and health care organizations can able to provide variety of on-demand services to the patients on cloud rather than owning standalone applications on local servers to reduce their cost in terms of purchasing resources and their maintenance.

3.3.4 Threat scenario

Patient information is stored on the cloud and trying to access with mobile health care applications at that time there is a possibility that any malicious program or administrator of service provider may misuse patient information, because, application requested query is operated on plain information of patient which is stored on the cloud database.

3.4 Health industry and Information Privacy

Patients are increasingly choosing for flexible timings and timely availability of medical consultation. For this, people tend to seek the required consultation through online health information systems. In this process they have to share their personal and health information online with third party health information systems which subsequently raises concerns over the privacy of patient related information. Information privacy [34] or data privacy is the relationship between collection and dissemination of data, technology, the public expectation of privacy, and the legal issues surrounding them. Every day internet users are sharing lot of information with each other through health care information system applications which are hosted on third party servers. In security context, Personal Identifiable Information (PII), refers to information that can be used to uniquely identify, contact or locate a single person or can be used with other sources to uniquely identify a single individual [35]. Sharing PII or even a combination of various pieces of information which leads to uniquely identifying the individual's poses a potential threat. Also from the business point of view, information privacy in cloud computing plays a crucial role. Therefore, cloud providers must ensure privacy of user information against curious administrators, malicious programs etc. According to the results of the Ponemon Institute and TRUSTe's 2008 Most Trusted Companies for Privacy Survey, "privacy is a key market differentiator in today's cyber world". There are also some information privacy protecting acts in place for instance Health Insurance Portability and Accountability Act(HIPAA), Gramm-Leach-Bliley Act (GLBA) and the FCC Customer Proprietary Network information (CPNI). As per these acts, the service providers need to protect privacy of individual's information.

3.5 Online Social Networks (Social web application)

Knowingly or unknowingly users of social networking sites give surprising amount of personal information freely and online Social Network service (for example: Facebook, LinkedIn, Twitter, Orkut) providers needs to store this vital information. Since Social Networking websites such as Facebook, Twitter have been growing rapidly with over two billions users now [50], Therefore majority of the social network sites providers choose to host their applications on public/private clouds. This is because of enormous advantages in term of cost and scalability.

3.5.1 Threat Scenario

A general fear concerning social networking sites is protecting the privacy of individuals and the sensitive information of others with whom they are sharing information. There have been articles [45] in the newspapers about social networks being misused because social networks have become the most popular platform for information sharing domain. These information sharing should be happened between intended users, unfortunately, it is not happening. If the Social network sites are hosted on the cloud, the administrator of this cloud may share customer's vital information for some monitoring benefits and vulnerable insider or malicious program in the cloud may misuse this information. Another threat is malicious third party applications accessing user information from the service provider's premises, example, horoscope application access user date of birth but how to ensure/guarantee that horoscope application is only accessing date of birth of user nothing else? also what is the guarantee that this application does not abuse accessed user information?

3.6 E-Governance

E-Governance is becoming a very important tool for the government to provide the services to the citizens in a handy, proficient and transparent way. According to the World Bank [10], E-Governance refers to the use by government agencies of information technologies (such as Wide Area Networks, the Internet, and mobile computing) that have the ability to transform relations with citizens, businesses, and other arms of government. These technologies can serve a variety of different ends: better delivery of government services to citizens, improved interactions with business and industry, citizen empowerment through access to information, or more efficient government management. The resulting benefits can be less corruption, increased transparency, greater convenience, revenue growth, and/or cost reductions. Today most of the countries are facing financial crises and as a result they are cutting extra spending and as a result governments are shrinking ICT budget [57]. Cloud computing is potential machinery that present a better solutions for e-governance. It provides service oriented architecture with lot of financial benefits in many aspects compared with on site usage of computational and storage resources. But when using cloud computing there are some potential threats associated with it.

3.6.1 Threat Scenario

Information stored in third party servers generally placed in a multitenant environment, where resources are being shared with other customers of the cloud providers. Therefore, organizations especially, government should be very careful storing the regulated and insightful citizen's information in cloud computing. The problems with shares and multi- tenancy resources in cloud computing are isolation mechanism within resources and across cloud service providers. A failure isolation mechanism becomes a potential threat for sensitive and regulatory information of citizens.

3.6.2 E-voting threat scenario

There have been many cases regarding the election rigging, election booth capturing, threatening of voters and many more situations in election system. The usage of E-voting system definitely reduces the misuse of votes in election system. But, consider a case, where the booth in charge or administrator of the electoral system is corrupted or forced to do some malpractice, then, e-voting system may compromise. It is possible only when he knows the status of polling completely, including how many votes each candidate got. Even, if it is connected to centrally located server, what is the guarantee that central server administrator cannot misuse if he knows the status. Therefore, except percentage of voting, nothing should be known to anybody at any level. Further, in [33] authors pointed out inside risks in an independent assessment of the voting systems such risks are: a malicious code might make use of vulnerabilities in the voting software to extend damage from machine to machine, an attacker could tamper with an e-voting machine while it is stored unattended overnight in a polling place, for some of the systems a voter could introduce malicious code in under a minute while voting, buffer overrun vulnerabilities and flawed cryptography algorithms, encryption keys hard-coded in the source code(the keys are the same for all machines using that software) and many more. This assessment clearly shows that even e-voting systems may have flaws they could be anything. All the votes must be in encrypted form at all the times, including, at the time of computation like counting and searching at booth level and central. Therefore confidentiality has to maintain at all the levels in the e-voting systems.

3.7 E-Mail System and threat scenario

Did you ever think that how much secure your email account? What is the guarantee that your composed mail is not seen by anybody (Email service providers)? One important concern with email system is that user simply trusts the e-mail service providers. Majority of the email service providers are processing the individual emails for filtering them. Example, spam emails and scan for any malwares. While processing email, if it is in encrypted form then they are decrypting the email. Another scenario is searching with keyword in the mail, where encrypted emails are decrypted and search takes place. In these scenarios emails of individuals are exposed to the service providers, which is a potential threat.

3.8 Other possible threats in the cloud service provider's environment

Apart from the threat scenarios discussed above with respect to each application, the following threats are also possible in third party services:

3.8.1 Side channel attack

In cryptography, a side channel attacks any attack based on information gained from the physical implementation of a cryptosystem, rather than brute force or theoretical weaknesses in the algorithms (compare cryptanalysis) [7].These attacks may be based on power consumption, timing information, electromagnetic leaks, Differential fault analysis and may be with sound can give some source of data that may be subjugated to crack the system. Even though servers of service provides are totally secured and the operating environment (OS, virtual machines, database, entire platform where our code is going to run) is completely trusted, still side channel attacks are capable of compromising the security of the servers. For example, an attacker using the cold boot attack [46] is able to retrieve sensitive data from the unrepressed DRAM after using a cold reboot to restart the machine. Using the branch prediction attacks [47] an attacker can collect information in relation to the encryption keys by simply monitoring the CPU time. These attacks are possible only when an attacker able access the system physically. In [53] authors have provided the Overview of cache-based side channel attacks: UP, SMT and SMP stand for uniprocessor, simultaneous multithreading and symmetric multiprocessing, respectively. The shown cache-based side channel attacks are against AES, DES, and RSA. This kind of side channel attacks may steal the decryption keys from the service provider systems, if the service provider system compromises, the attacker can easily steal the information and uses these keys for decryption of the encrypted information. Therefore, there must be a way that even side channel attacks need to be tackled in efficient way in the cloud service environment.

3.8.2 Hardware Trojan Horses (HTH)

It is very hard to detect hardware Trojans Horses in the system. The Service providers sometimes may not have taken prevention from hardware Trojans attacks. Even if service providers are taking help of TPM or SoC to protect keys, in the presence of hardware Trojans, it is very risky. The competitors or corrupt designers or corrupt fabricators of IC may intentionally keep hardware Trojans to steal the keys or sensitive information stored in the cloud. Therefore, user of the cloud should not share keys with the service providers. Hardware Trojan concerns have been documented in reports from the US Defense Science Board task force [37], the US Senate [38], IEEE Spectrum [39], and Semiconductor Equipment and Materials International (SEMI)[40].

3.8.3 Zero-Day Attacks

A zero-day (or zero-hour or day zero) attack or threat is an attack that exploits a previously unknown vulnerability in a computer application, meaning that the attack occurs on "day zero" of awareness of the vulnerability [58]. Service providers of cloud may have countermeasure for known vulnerability, if zero-day attack takes place in the system of service providers, this may steal sensitive information of the cloud users. If the attack occurs at the time of processing of the data and if data is plain then this information will be under threat. While processing if decryption keys are used then these keys are also under threat.

4. TARGET SECURITY IN THIRD PARTY SERVICE PROVIDER'S ENVIRONMENT (CLOUD COMPUTING)

We have discussed security concerns in cloud computing under the section three with respect to social and business applications. The following section describes how to achieve confidentiality in the face of security concern or threats in third party services (Cloud Computing) in general scenario with respect to a search operation. i) Consider all of our information/data stored in cloud in encrypted format and if we want to search an encrypted search terms in the cloud for specific information. ii)Let information in cloud database and search term be in encrypted format and

search encrypted search keyword directly in the still-encrypted database, and get the same results. iii) If we can perform calculations directly on our encrypted data and yet get the same results that we get from the unencrypted data. iv) This scenario provides a win-win situation to both service providers and service users enormously from the security and privacy point of view. v) With this scenario we need not required sharing our decryption keys with service providers, therefore, no need to trust service providers to steal or sell, lose of our data. vi) Since there are no decryption keys are with service providers, service provider or any malicious programs or any other kind of mechanism can't see or misuse of our data even though they wanted to. This kind of security in the cloud computing environment can be achieved with homomorphic encryption. Where query and information are always in encrypted form and encrypted query is operated directly on encrypted information without decrypting either query or information. All the cryptographic operations are performed at the user end only. Therefore, no need to share the keys with the service provider hence no fear that service provider may misuse information stored in the cloud. However, this homomorphic encryption ensures only confidentiality of the information stored in the cloud; it does not guarantee other aspects like integrity, availability of information. Another scheme which allows performing operation on encrypted data is CryptDB, where as CryptDB is a scheme which supports confidentiality from malicious administrators and malicious programs for applications backed by SQL databases.

5. ANALYSIS OF APPLICABILITY OF HOMOMORPHIC ENCRYPTION AND CRYPTDB IN SOCIAL AND BUSINESS APPLICATIONS IN CLOUD COMPUTING ENVIRONMENT

5.1 Financial Institutes

Homomorphic encryption helps in a great way to the bankers to think of going for the cloud based solutions. Because, in homomorphic encryption technique information of customers account will be in encrypted form, third party service provider cannot get any useful information to misuse while computation. The encryption and decryption takes place at bank and still cloud does the useful computation on encrypted data without decrypting it. Transactions like deposit and withdraw of money require addition and subtraction operation. For example, if balance amount of customer is encrypted with homomorphic encryption, it is very much possible to do deposit transactions on encrypted balance field itself without decrypting it. Since, balance is field is in encrypted form bank employee cannot see clear text of it, therefore chance of misusing is very less (Informing balance of customers to others).Ultimately, a user or the bank prefers a kind of way out which did not involve decryption of user's data by the third party service providers even while processing it. The banker can outsource all of his computational as well as storage requirements and still can able to maintain the confidential of the customer's data. This way homomorphic encryption is very much applicable in banking application to protect customer's information from spurious administrators of service providers and internal employees.

5.2 University Examination System

To mitigate problems (threats) associated with university examination cell, homomorphic encryption mechanism may be used, because, question papers and marks are in encrypted form and corrupt administrator can't see the original marks list and all required operations are performed on encrypted marks list itself. The authorized person will enter the marks of the students through secured interface. All the entered marks are homomorphically encrypted before storing into the examination server database or any third party service environment. Applications access only encrypted marks from cloud/server and required manipulations (if any) are done in the cloud on encrypted marks only, except at the time of printing out marks sheets, the marks remains in encrypted form in the database. Mostly, search, sort, addition, subtraction and multiplication operations are required to be performed on marks list. Homomorphic encryption and CryptDB supports all these operations, hence, homomorphic encryption and cryptDB are very much applicable in university examination system to protect the confidentiality.

5.3 Online Social Networks (Social web application)

When user is creating a new account with social web sites through their application interface, there must be a plug-in with application which encrypts all the user information and submits. Subsequently, the correspondences such as searching for a specific information and updation of information are also need to be encrypted before submission and do not share keys with service providers and decryption will be done at receipt end .This ensures two things first, on transition data is secured,
second, since we are storing and processing all the possible operation on encrypted data, confidentiality of user information is maintained. This can be achieved with CryptDB where cryptDB supports searching and updation operations on encrypted information. Therefore, CryptDB is applicable in protecting the confidentiality of user information.

5.4 Mobile health care and Healthy industry

To maintain absolute confidentiality of patient's information, this information should be in encrypted format so that only intended user or authorized person can view this data for any computations or just to see. Mobile health care applications will access the encrypted patient information from cloud and decrypt at their end. If application requires any computation (searching, sorting, addition, etc...) on patients information, all these computations are done at service provider's environment
on encrypted patient information only. Therefore, any malicious administrator, malicious programs, cannot able to see patients information in plain form. Homomorphic encryption and CryptDB schemes support searching, sorting, addition, etc.. Operations on encrypted patient's information, hence homomorphic encryption and cryptDB schemes are applicable to protect the confidentiality of patient information in the cloud.

5.5 E-Governance

Citizens sensitive information before hosting on cloud server need to be homomorphically encrypted and any computations (Searching, sorting, addition, multiplication, etc..) if required for attending the service need to be done on encrypted data itself in the cloud. Allow only authorized person at the recipient (Individual, government e-service center) can able to decrypt information. This can be achieved with homomorphic encryption and CryptDB schemes. Depending on the requirement, these schemes may be used.

5.5.1 E-voting

After voter casts his vote, this information has to be homomorphically encrypted and store in the local database or central database and subsequent voter casting also encrypted and added to the respective encrypted fields in the database (local or central). Since, homomorphic encryption supports addition operation on encrypted voting information, then there is no

requirement for decrypting voting information for performing addition operation either locally or central. Therefore, homomorphic encryption can be used in the e-voting system for providing the confidentiality. This technique can be used at the time of casting and counting the votes. Since, voting information is in encrypted format malicious local or central administration or manager cannot get the status of voting information. Even any malicious program presents and read the information; information will be available in encrypted form only.

5.6 Mobile Commerce

All the sensitive fields of user information and with any combination user personal identification is possible are also need to be homomorphically encrypted and all possible operation needs to be done on encrypted information. So that, any malicious administrator and internal employees cannot see the user vital information. Homomorphic encryption and CryptDB supports performing operation on encrypted data itself.

5.7 E-Mail System

It would be very nice that if we encrypt the email subject and still able to perform search operation on encrypted email content with an encrypted search keyword. Then email users are need not worry about the confidentiality of their email information even though their emails are being hosted on the cloud or any third party services. Essentially, in the service providers environment operations on our email account should be done on encrypted data only. CryptDB supports the searching operation on encrypted data itself.

5.8 Side channel attack

One of the best possible solutions in the face of side channel attack is homomorphic encryption. Even the attacker gains access to the service provider system by means of side channel attack, he cannot understand anything because all the information in the system is in encrypted form. Even attacker gains the root privileges he cannot able to decrypt the information because the decryption keys are with the user and never are they shared with service provides. Since they are no keys are available with the compromised system, decryption is becomes almost impossible. This kind of security is possible when we use the homomorphic technique. All the information/data are to be encrypted before sending to cloud/third party service. Therefore, Homomorphic encryption is a potential solution against the side channel attacks to preserve the confidentiality of the user data which is stored in the cloud.

5.9 Hardware Trojan Horses (HTH)

Under this kind of threat scenario in cloud computing, homomorphic encryption mechanism is a better option for encrypting the sensitive information before storing in the cloud. Since keys are not with the service provider's domain and no point in time our data will be in plain text, the presence of Hardware Trojans does not see original values of our data and hence confidentiality of information will be preserved even though hardware Trojans steal user data and send to their originators.

5.10 Zero-day attack

To maintain the confidentiality of the information in the face of zero day attacks in the cloud computing, information never should be in plain format and also keys should not be shared with the service providers. Since, Homomorphic encryption scheme supports processing on encrypted data, no requirement to decrypt information to plain text to process it. Hence, any kind of zero-day attacks cannot read the information, because, information always remains in encrypted format only. Therefore, in the face of zero-day attacks, confidentiality of the data is ensured with homomorphic encryption. In the following section described some of the customer information privacy acts and applicability of homomorphic encryption to protect customer's information privacy.

6. HOMOMORPHIC ENCRYPTION

The main concept of homomorphic encryption is to protect information from unauthorized users by supporting computation on encrypted data. Like any other traditional cryptography, main aim is to maintain confidentiality of user information at transmission, storage and also additionally at processing time. It supports computation, searching etc mechanisms on encrypted data without requiring information to be decrypted and used at the time of processing. There are some partial homomorphic encryption techniques available in the literature, but major breakthrough was in the year of 2009 by the Gentry a Stanford university PhD research scholar. One of the advantages with homomorphic encryption is no need to decrypt data for processing; this saves the decryption time while processing large set of data. In addition, no need to share the keys, this eliminates threats of key exchange. The mathematical notations of homomorphic encryption are shown below: Where M is message to be encrypted, c= E_k(M), Encryption of message M with key k, M=D_k (c), decryption of cipher text c into plain message M with key k. Now, let us considered the homomorphic property: C1= E_k(M), C2=E_k (M1). Now, able to perform arbitrary computations on C1 and C2, if C1+C2 then it called additive homomorphic encryption, i.e. C3= C1+C2. If perform C1*C2 then it called multiplicative homomorphic encryption, i.e. C4=C1*C2. The decryption result of C3, C4 is sum of M1, M2 and product of M1, M2 respectively. For any given data it possible to compute the arbitrary number of computations on encrypted data is the main concept of homomorpic encryption. The operation such as addition and multiplication on encrypted data is analogous to same operations on plain data. Homomorphic encryptions are two types; one is partial homomorphic encryption and second is full homomorpic encryption. The partial homomorphic encryption supports only either addition or multiplication, where full homomorphic encryption supports both operations such as addition and multiplication on encrypted data which supports ring structure. The figure-1 illustrates that an application sends an encrypted query(homoquary) to cloud and receives the encrypted results(homoreply).

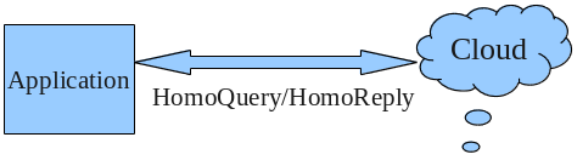


Fig. 1 Homomorphic encryption Based Cloud architecture

6.1 Partial Homomorphic Encryption Algorithms

Under this section, discussed some of the partial homomorphic encryption algorithms and implemented four partial homomorphic encryption supported algorithms and the results and performance are discussed in details.

6.1.1 Paillier Algorithm

The Paillier algorithm [52] is invented by Pascal Paillier in 1999 and it is a probabilistic asymmetric algorithm for public key cryptography. The problem of computing nth residue classes is believed to be computationally difficult. The decisional composite residuosity assumption is the intractability hypothesis upon which this cryptosystem is based. The scheme is an

additive homomorphic cryptosystem where it is possible only to perform addition operation on encrypted data. By multiplying the two encrypted messages will be resulting two addition of their plain messages mod n. D[E(m1).E(m2)mod n2]=(m1 + m2) mod n. With this property one can use in the applications where it required to perform addition operation on sensitive data securely. The application such as E-voting systems, banking, university examinations cell and wherever application required perform sum computations on data.

6.1.2 RSA

Ron Rivest, Adi Shamir and Leonard Adleman developed a public key encryption scheme called RSA [8]. It was one of the very earlier multiplicative homomorphic encryption schemes. *Multiplicative homomorphic property of RSA:*

If c1 = M_1^e mod n, c2 = M_2^e mod n, then c1 × c2 = (m1 × m2) ^{e} mod n is multiplicative homomorphic property of RSA can be used in many applications where those applications required performing multiplicative operations on sensitive user data.

6.1.3 Elgamal

ElGamal encryption was introduced by T. ElGamal in 1985 [18]. ElGamal is a public key crypto algorithm. Input X and a random number r produce an encrypted pair E(x,y) based on a public key {p,t,B} and can then be decrypted using a private key. This encrypted pair is multiplicatively homomorphic. In the ElGamal cryptosystem, in a group G, if the public key is (G,q,g,h), where h= g^x, and x is the secrete key, then the encryption of a message m is E(m) = (g^r, m . h^r), for some random $r \in \{0, \ldots, q-1\}$. The homomorphic property is then :E(x1) * E(x2) = (g^{r1}, x1,h^{r1}) (g^{r2}, x2,h^{r2}) = = (g^{r1+r2}, (x1 * x2)h^{r1+r2}) = E(x1*x2).

6.1.4 Benaloh

Benaloh [42] is public key cryptosystem having additive homomorphic property. In the Benaloh cryptosystem, if the public key is the modulus m and the base g with a block size of c, then the encryption of a message x is $\varepsilon(x) = g^x r^c$ mod m, for some random r $\in \{0, \ldots, m-1\}$.

The homomorphic property is then $\varepsilon(x1).\ \varepsilon(x2)$ =($g^{x1} r_1^c$) ($g^{x2} r_2^c$) = g^{x1+x2} $(r_1\ r_2)^c$ = $\varepsilon(x1 + x2\ mod\ c)$.

6.1.5 Empirical results of Paillier, RSA, Elgamal and Benaloh Partial Homomorphic Encryption Algorithms

Implemented, tested and presenting the results of above four algorithms on three different fully virtual machines and host machine. The configuration of host machine is i7 processor, 8GB RAM and 2TB hard disk with Fedora Linux kernel 2.6.35.13-92. The virtual machines configurations are provided in Table -1. The virtual machines in our implementations are VirtualBox, VMPlayer and QEMU/KVM. The Guest Operating system is Opensuse Linux kernel .6.37.1-1.2. The algorithms implemented are Paillier, Elgamal, RSA and Benaloh. We have considered two additive homomorphic encryption supported algorithms (Paillier, Benaloh) and two multiplicative homomorphic supported algorithms (Elgamal, RSA).For the algorithms, Paillier, Elgamal, RSA, we have taken one lakh, 2 ten digit numbers and calculated the average timings of encryption, addition of encrypted data, multiplication of encrypted data, decryption of added encrypted data, decryption of multiplied encrypted data for one lakh 2 ten digits numbers. We have also calculated timings for storing and retrieving one lakh ten digits plain data to and from remote database. We have also calculated the average storing timings of one lakh two ten digits encrypted data onto remote database and also calculated retrieving timings of these one lakh encrypted records from remote data base. For Benaloh algorithm, we have considered two one lakh 2 digits numbers only and calculated timings as mentioned above. The remote data base is located within the same subnet of local system. The Paillier algorithm provides a partial multiplication homomorphic property, while multiplying of two numbers first number is encrypted and multiplied with unencrypted second number. The detailed calculation timings are given in tables from 4-11with respect to each algorithm. The inputs for Paillier, Elgamal, RSA is two one lakh ten digit numbers, two numbers are chosen because need to perform addition and multiplication with these algorithms. The input for Benaloh algorithm is one lakh two digit numbers, in this case only two digits are chosen because this algorithm was taking lot of time with more than two digits, this algorithm considered to be very expensive in terms performance. Figure-2 illustrates the full virtualization setup of our implementation using three full virtual machines such as VirtualBox, VMPlayer and QEMU/KVM. We have considered these hypervisors because of their availability in the open source and need not required to change the underline host operating system. Input data for the three algorithms (Paillier, Elgamal, and RSA) are shown in table-2 and input data for Benaloh algorithm is shown in table-3.

Table 3: Input data for Benaloh algorithm

Sno	Number-1 (Random 2 digits)	Number-2 (Random 2 digits)
1	56	23
---	----	-------
100000	98	47

Table 2: Input data for (Paillier, Elgamal, and RSA) algorithms

Sno	Number-1 (Random 10 digits)	Number-2 (Random 10 digits)
1	1234534566	1234589605
---	----	-------
100000	2347890457	1456789045

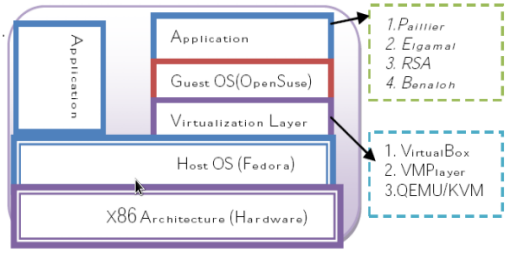


Fig. 2: Virtualization Setup for Testing the Algorithms

Initially, we have stored one lakh two ten digits numbers in mysql database and retrieved and encrypted these two one lakh ten digits numbers separately and calculated their average encryption timings for these one lakh records. Next, performed the addition and multiplication operating on encrypted data and calculated average addition and multiplication timings. Similarly, calculated average decryption timings for the added and multiplied encrypted data. Average in the result means average of 1 lakh records. All the results are in nano seconds (ns). One of the reasons that performance of these algorithms

Table 1 Specifications of virtual machines

Specification	VirtualBox	VMPlayer	QEMU/KVM
No. of CPUs	2 VCPU	2VCPU	2 VCPU
RAM	2GB	2GB	2GB
Host OS	Fedora	Fedora	Fedora
Guest OS	Opensuse	Opensuse	Opensuse
Network configure	Attached to : Bridged adapter	Attached to : Bridged adapter	Attached to : NAT
Network Bandwidth	10/100 Mbps	10/100 Mbps	10/100 Mbps
Harddisk	15GB	15GB	15GB
Database	Mysql	Mysql	Mysql
IDE	NetBeans-7.4	NetBeans-7.4	Netbeans-7.4
Language	Java	Java	Java

may be varying depending on the key sizes considered, apart from their implementations on different virtual machines. In Paillier crypto system, generated two randomly positive Big Integers that are probably prime, with the specified bit Length 512 and certainty 64 and generated two prime numbers are used to generate public key. For RSA, public key chosen is 65537 and private key is 553334639749230512972091308 9. For Elgamal, private key is 1234567890 and public key is 7638435862451621731. For Benaloh, private keys are two big prime numbers p and q and public keys parameters are y,n;p and q are 156 digits big integers and y and n are 312 digits big integers. The complete results and overheads calculations, overhead comparison (timings) are given in the tables from 4-18.The sizes of ciphertext after encryption, addition and multiplication operation on encrypted data are also show in the table-19 for four algorithms. From tables 8-11 results are based on database hosted on remote machine. Graphs from 1-5 shows the comparison of overhead with encrypted data over plain data for addition and multiplication operations for four algorithms. The following abbreviations are used in the tables from 4-11. VM: Virtual Machine; HM: Host Machine; AET2: Average Encryption timing for two one lakh 10-digit numbers (ns); AA2E: Average addition of two one lakh 10- digits numbers encrypted data timings (ns); AD2E: Average decryption timings of added 2 one lakh 10-digits encrypted data (ns); AM2E: Average multiplication of two one lakh 10- digits numbers encrypted data timings (ns); AD2ME:Average Decryption timings of multiplied 2 one lakh 10- digits encrypted data (ns).

Table 4. Results of Paillier Algorithm on Host machine and on different Virtual Machines

VM/HM	AET2	AA2E	AD2E	AM2E	AD2
HM	4637899	7399	4123558	143725	4106848
Virtual Box	5763389	7712	5102817	179993	4978856
VMPlayer	3940770	6724	3694068	128467	36863
QEMU/KVM	3945544	6797	3699467	128789	696941

Table 5. Results of Elgamal Algorithm on Host machine and on different virtual machines

VM/HM	AET2	AM2E	AD2ME
HM	37441.1793	889.5170	18722.518
Virtual Box	22493.7547	728.2301	12490.004
VMPlayer	16706.8507	2921.3812	8582.7304
QEMU/KVM	17196.1173	3311.8602	12159.750

Table 6. Results of RSA Algorithm on Host machine and on different virtual Machines

Virtual Machine/ Host machine	AET2	AM2E	AD2ME
HM	16881.7341	930.3111	42983.0742
Virtual Box	12783.3587	635.8859	23223.2196
VMPlayer	10072.5500	863.8477	18462.7488
QEMU/KVM	11753.3289	562.0823	17735.1093

Table 7. Results of Benaloh Algorithm on Host machine and on different Virtual Machines

VM/HM	AET2	AA2E	AD2E (ms)
HM	392913.5629	2380.7525	2341227386004
Virtual Box	491548.3535	2752.5567	11242102828681
VMPlayer	321855.7547	1638.1649	5293974487839
QEMU/KVM	355669.8346	1991.4468	5469413799581

Table 8. Paillier Algorithm

VH/HM	One lakh 10-Digits Plain data storing onto Remote Database Timings(ns)	One lakh 10-digit Plain Data Retrieving from Remote Database Timings(ns	Average two 10-digits Encrypted data storing timings on remote Database(ns)	Two one lakh Encrypted data retrieving timings from remote database(ns)
HM	79602938359	244758194	1007085	5352871322
VirtualBox	11033023546	33218317	1116383	5560466914
VMPlayer	92647282758	29774894	1103970	5670430487
QEMU/KV	M118758064	28011704	1370803	5596187770

Table 9. Elgamal Algorithm

VM/HM	Average two one lakh ten digit numbers encrypted data storing timings(ns)	Retrieving two one lakh encrypted data timings (ns)
HM	778212.2146	376793541
Virtual Box	1113011.3910	401117791
VMPlayer	934501.4437	378696177
QEMU/KVM	1205078.1696	393303481

Table 10. RSA Algorithm

VM/HM	Average two one lakh ten digit Numbers Encrypted data storing timings(ns)	Retrieving two one lakh ten digit encrypted data timings(ns)
HM	811439.2649	563870998
Virtual Box	1119615.4912 1	572365618
VMPlayer	933226.9286	556169480
QEMU/KVM	1229763.2439	567895853

Table 11. Benaloh Algorithm

VM/HM	Average two one lakh ten digit numbers encrypted data storing timings(ns)	Retrieving two one lakh ten digit encrypted data timings (ns)
HM	1061823.9314	5749744411

Virtual Box	1265576.2116	5349154763
VMPlayer	1104598.3080	5716719636
QEMU/KVM	1563666.8861	5975955788

Table 12. Average timings for two one lakh ten digits plain data addition and multiplication

Platform	Average Addition of two one lakh ten digit numbers timing(ns)	Average multiplica-tion of two one lakh ten digit numbers timing(ns)
Host	424.2612	516.3624
VirtualBox	513.0177	451.9619
VMplayer	299.5329	248.4567
QEMU/KVM	567.0812	372.9772

Table 13. Comparison of Overhead with encrypted data addition over plain data for Paillier algorithm

Platform	Plain data addition(ns)	Encrypted data addition(ns)
Host	424.2612	7399.5395
VirtualBox	513.0177	7712.3639
VMplayer	299.5329	6724.2671
QEMU/KVM	567.0812	6797.4152

Table 14. Comparison of overhead with encrypted data multiplication over plain data for Paillier algorithm

Platform	Plain data multiplica-tion(ns)	Encrypted data mul-
Host	516.3624	143725.7328
VirtualBox	451.9619	179993.5427
VMplayer	248.4567	128467.3121
QEMU/KVM	372.9772	128789.3700

Table 15. comparison of overhead with encrypted data multiplication over plain data for Elgamal Algorithm

Platform	Plain data Multiplication(ns)	Encrypted data multiplication(ns)
Host	516.3624	889.5170
VBox	451.9619	728.2301
VMplayer	248.4567	2921.3812
QEMU/KVM	372.9772	3311.8602

Table 16. Comparison of overhead encrypted data multiplication over plain data for RSA algorithm

Platform	Plain data Multiplication(ns)	Encrypted data multiplication(ns)
Host	516.3624	930.3111
VBox	451.9619	635.8859
VMplayer	248.4567	863.8477
QEMU/KVM	372.9772	562.0823

Table 17. Comparison of overhead encrypted data addition over plain data for Benaloh algorithm

Platform	Plain data addition(ns)	Encrypted data addition(ns)
Host	424.2612	2380.7525
VirtualBox	513.0177	2752.5567
VMplayer	299.5329	1638.1649
QEMU/KVM	567.0812	1991.4468

Table 18. Consolidated overhead of four Algorithms homomorphic property over plain data in numbers of times

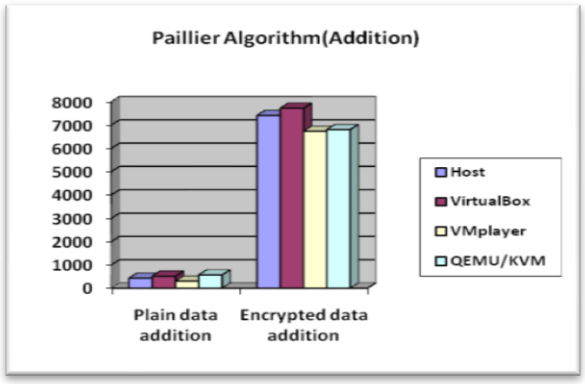
Platform	Algorithm	Over head Addition of Encrypted data over plain data for 10-digit numbers	Over Head Multiplication of encrypted data over plain data for 10-digit numbers
Host	Paillier	17 times	278 times
Host	Elgamal	NA	1.72 times
Host	RSA	NA	1.79 times
Host	Benaloh	5.6 times	NA
Virtual Box	Paillier	15 times	399 times
Virtual Box	Elgamal	NA	1.6 times
Virtual Box	RSA	NA	1.5 times
Virtual Box	Benaloh	5.36 times	NA
VMPlayer	Paillier	22.4 times	518 times
VMPlayer	Elgamal	NA	11.7 times
VMPlayer	RSA	NA	3.48 times
VMPlayer	Benaloh	5.48 times	NA
KVM	Paillier	11.99 times	346 times
KVM	Elgamal	NA	8.9 times
KVM	RSA	NA	1.5 times
KVM	Benaloh	3.5 times	NA

Table 19. Length of cipher texts for four Algorithms after encryption, addition and multiplication

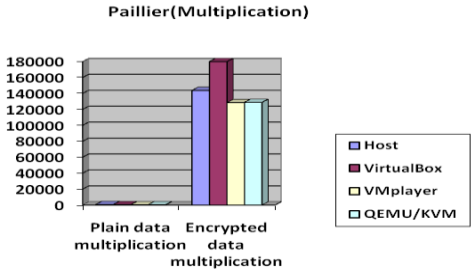
Algorithm	Length of cipher text for ten digit number after encryption(bits)	Length of cipher text after addition(bits)	Length of cipher text after multiplication(bits)
Paillier	1020	1021	1021
Elgamal	64	NA	64
RSA	99	NA	198
Benaloh	1022	2044	NA

Table 20. Partial homomorphic encryption algorithms and their homomorphism property

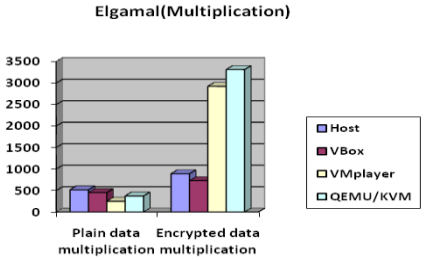
Sno	Algorithm	Homomorphism
1	Paillier ECC variations [54]	Additive
2	Naccache-Stern [49]	Additive
3	Okamoto-Uchiyama [9]	Additive
4	Kawachi-Tanaka-Xagawa [55]	Additive
5	Melchor-Gaborit-Herranz [3]	Additively with d-Operand Multiplications
6	Damgard-Jurik [4]	Additive
7	Goldwasser-Micali [43]	XOR



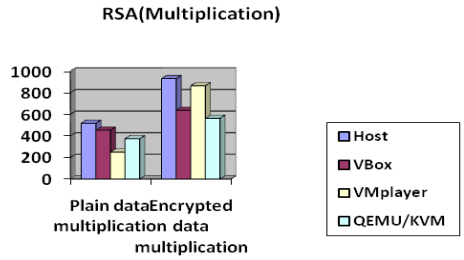
Graph-1: Comparison of overhead with encrypted data over plain data for addition operation for Paillier Algorithm(y-axis in nanoseconds)



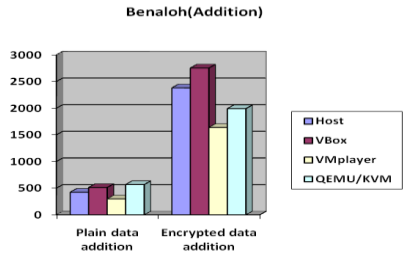
Graph-2: Comparison of overhead with encrypted data over plain data for multiplication operation for Paillier Algorithm(y- axis in nanoseconds)



Graph-3: Comparison of overhead with encrypted data over plain data for multiplication operation for Elgamal Algorithm(y- axis in nanoseconds)



Graph-4: Comparison of overhead with encrypted data over plain data for multiplication operation for RSA Algorithm(y- axis in nanoseconds)



Graph-5: Comparison of overhead with encrypted data over plain data for addition operation for Benaloh Algorithm(y-axis in nanoseconds)

The results of these algorithms clearly shows that these algorithms are very much practical and they can be used in applications mentioned in this paper wherever they are applicable. The performance of Benaloh decryption timings of added encrypted data is much higher. The timings in this implementation are nano seconds except for the Benaloh algorithm decryption of added encrypted data and this is in milliseconds. The cipher text size of the algorithms is reasonable even after performing addition and multiplication operations on cipher text. The average timings for storing and retrieving the cipher text to and from remote database are recorded and their performance is very much reasonable, therefore do not have much band width problems. The shown overheads with four partial homomorphic encryptions are sustainable with the present microprocessor and network technology. Especially in the cloud computing where high speed CPUs and Network equipments are used. The table 20 lists some more partial homomorphic encryption algorithms and their homomorphism property.

Limitations of Partial homomorphic encryption schemes

Undoubtedly, the partial Homomorphic encryption mechanisms are helpful practically in many social and business applications. The partial homomorphic encryption algorithms are efficient and secure enough to use them in practical applications. The partial Homomorphic Encryption algorithms like Elgamal, Benaloh, RSA, Paillier are well proved in term of security therefore, one can consider these schemes for implementing practical applications. However, the partial homomorphic encryption schemes have some limitations, majority of these schemes support only one type of operation, therefore usage of these schemes in practical applications have big restriction, and most of the applications requires more than one operation need to be performed. Therefore, these algorithms need to be used in combination with other algorithms as required by the applications.

6.2 Fully homomorphic Encryption Algorithms

In this section, the state of the art fully homomorphic encryption schemes are discussed: In [16,17,19,20,21,22,23,24,25,26,27,28,29,30,31] authors have described various fully homomorphic encryption scheme, they are listed in the following table 21. In fully homomorphic encryption scheme, two operations are possible to perform on encrypted data such as addition and multiplication. Till 2009, there was not much work done on fully homomorphic encryption. Gentry [41], in his PhD thesis have proposed a fully homomorphic encryption scheme with lattices, this was a big breakthrough in the cryptography domain. To prove the practicality of homomorphic encryption efforts are made to build

the homomorphic encryption softwares. This resulted in two open source libraries. First one is Helib [14,15] and second is libScarab [44] .Efforts is made to design architecture to execute on encrypted data. In [13] authors are explained the designed principles of One Instruction Set Architecture for computing the encrypted data. This novel architecture combines the simplicity and high throughput of OISC with the security of well-known homomorphic encryption schemes, allowing execution of encrypted machine code and secure computation over encrypted data. In [13] the authors have explained the Paillier's homomorphic scheme for encryption using addleq instruction of OISC. Preliminary results in this implemented hardware cognizant software simulator indicate an average execution overhead of 26 times for 1024 bit security parameter, compared to unencrypted execution of the same OISC programs. The authors of [11] propose an approach for secret program execution, based on fully homomorphic encrypted circuits. On the theoretical front, the authors of [12] provide theoretical proof of the correctness of an encrypted processing unit.

Limitation of Fully HE Schemes

Compared to the partial Homomorphic schemes, Fully Homomorphic Encryption schemes are capable of support both additions and multiplication for limitless times. This feature permits processing of any Boolean circuits. However, one big question is efficiency of available fully homomorphic encryption schemes with respect to computational speed and size of the cipher text. Gentry and Halevi [48] scheme was taking more than a second for encryption of a single bit on high-end Intel Xeon based server and recryption primitive taking nearly 30 seconds for the lowest security setting. Recryption operation needs to be applied periodically for bit-AND operations to trim down the noise in cipher text to a controllable level. The Gentry-Halevi scheme requires a ciphertext of more than 780,000 bits for encrypting a single bit [6]. This enormous ciphertext size may become problem on bandwidth requirement to transmit the cipher texts. Still, fully homomorphic encryption has to be matured in terms of computational speed and size of produced cipher text. It becomes problem when need to store the cipher text in the database, some database system may not support a field to hold

very huge data in the table. However, they are practical with very high speed CPUs, particularly, in cloud computing, where high performance computers are being used for computational purposes.

6.3 The Somewhat Homomorphic Encryption Scheme

Gentry [41] proposed the first FHE scheme that supports performing an arbitrary addition, multiplication on encrypted data. To encrypt a bit b, the encryption formula is c= pubkey∗q+2∗r+b, where r and q are two random integers and pubkey is the public key and with condition that 2*r is smaller than pubkey/2. The decryption, b= c mod seckey mod 2, where seckey is correspondent secret key of public key. In fully Homomorphic Encryption, compute both sums and products of bits or numbers.

Secrete key somewhat homomorphic encryption
Secret key: large odd number p
To encrypt a bit b:
i) Pick a (random) "large" multiple of p, say q·p
ii) pick a (random) "small" number r (2.r+b) (this is even if b=0, and odd if b=1). iii)Cipher text c =q·p+2·r+b
To decrypt a ciphertext c: Taking c mod p recovers the noise

Plaintext =c mod p mod 2. If there is no noise i.e. r=0, with GCD we can find out secrete key P. Therefore ensure the noise (r) should not be 0. Now, let us perform the Exclusive OR with two encrypted bits,m1 & m2.$c_1 = q_1 \cdot p + (2 \cdot r_1 + m_1)$, $c2 = q_2 \cdot p + (2 \cdot r_2 + m_2)$, $c_1+c_2 = p \cdot (q_1 + q_2) + 2 \cdot (r_1+r_2) + (m_1+m_2)$ Odd if $m_1=0$, $m_2=1$ (or) $m_1=1$, $m_2=0$; Even if $m_1=0$, $m_2=0$ (or) $m_1=1$, $m_2=1$ $c_1+c_2 = p \cdot (q_1 + q_2) + 2 \cdot (r_1+r_2) + (m_1+m_2)$, the LSB of equation is(m_1 XOR m_2),Now, let us perform the ANDing with two encrypted bits, m1 & m2. $c_1 = q_1 \cdot p + (2 \cdot r_1 + m_1)$, $c_2 = q_2 \cdot p + (2 \cdot r_2 + m_2)$, $c_1c_2 = p \cdot (c_2 \cdot q_1 + c_1 \cdot q_2 - q_1 \cdot q_2) + \underline{2 \cdot (r_1r_2 + r_1m_2 + r_2m_1) + m_1m_2}$, The LSB of equation is (m_1 AND m_2). The following two equations clearly show how noise is growing with respect to addition and multiplication operation $c_1+c_2 = p \cdot (q_1 + q_2) + \underline{2 \cdot (r_1+r_2) + (m_1+m_2)}$ Noise(r)= 2 * (initial noise), $c_1c_2 = p \cdot (c_2 \cdot q_1 + c_1 \cdot q_2 - q_1 \cdot q_2) + \underline{2 \cdot (r_1r_2 + r_1m_2 + r_2m_1) + m_1m_2}$, Noise(r) = (initial noise) ^{2}. The problem with this approach is adding doubles and multiplication squares the noise. If the noise is greater than p then decryption will output an incorrect bit. Even with this somewhat homomorphic encryption one can do lots of additions and some multiplications. This may be used in database searches applications and e-mail spam filtering. Gentry's "bootstrapping method" can be used to reduce the noise to a fixed level, whenever noise increased behind the certain limit. The encrypted secrete key is being used for fixing the noise level. The Josh's system supports many add operations and zero multiplications, where as Boneh, Goh & Nissim system supports many add operations and only one multiplication. Whereas the Gentry's somewhat homomorphic encryption supports fully homomorphic encryption with many add operations and many multiplication operations

Table 21. Fully Homomorphic Encryption Schemes

Sno	Name of scheme	Theme of Scheme
1.	ON DATA BANKS AND PRIVACY HOMOMORPHISMS	Different (basic) logics to perform addition, subtraction, multiplication on encrypted data
2.	Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP	A new tensoring technique for LWE-based fully homomorphic encryption. With every multiplication the cipher text noise grows linearly ($B \rightarrow B \cdot poly(n)$). Where as in other schemes the cipher text noise grows quadratically ($B \rightarrow B^2 \cdot poly(n)$) with every multiplication (before "refreshing"). This is a scale-invariant fully homomorphic encryption scheme, whose properties only depend on the ratio between the modulus q and the initial noise level B, and not on their absolute values.
3.	On-the-Fly Multiparty Computation on the Cloud via	This is a multikey Full Homomorphic Encryption, which is capable of

	Multikey Fully Homomorphic Encryption	operating on inputs encrypted under multiple, unrelated keys. A ciphertext resulting from a multikey evaluation can be jointly decrypted using the secret keys of all the users involved in the computation. This scheme is based on the NTRU.
4.	Homomorphic Evaluation of the AES Circuit	A leveled homomorphic encryption that evaluates AES-128 circuit. This technique is very much useful in application scenarios like when data is encrypted under AES and wanted to compute on that data, then homomorphic AES decryption would transform this AES-encrypted data into an FHE-encrypted data, and then we could perform whatever computation we wanted
5.	Fully Homomorphic Encryption without Squashing Using Depth-3 Arithmetic Circuits	This is a mechanism to construct a FHE as a hybrid of a Somewhat Homomorphic Encryption (SWHE) and a multiplicatively homomorphic encryption (MHE) scheme, such as Elgamal. This mechanism eliminates the requirement of squashing step and thereby also removes the need to assume the SSSP is hard.
6.	Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based	This is FHE based on the Learning with Errors problem. The technique to construct FHE is called approximate eigenvector method. In this scheme, for the most part, homomorphic addition and multiplication are just matrix addition and multiplication. This is known as first identity-based FHE scheme.
7.	Efficient Fully Homomorphic Encryption from (Standard) LWE	A fully homomorphic encryption scheme which is based solely on the (standard) learning with errors (LWE) assumption and a new re-linearization technique.
8.	Public Key Compression and Modulus Switching for Fully Homomorphic Encryption over the Integers	This is a compression technique that reduces the public key size of van Dijk, Gentry, Halevi and Vaikuntanathan's (DGHV) fully homomorphic scheme over the integers from O(λ^7) to O (λ^5).
9.	Fully Homomorphic Encryption with Polylog Overhead	The authors present a construction of fully homomorphic encryption (FHE) schemes that for security parameter λ can evaluate any width-Ω (λ) circuit with t gates in time $t \cdot$ polylog(λ) by using of batch homomorphic evaluation techniques of Smart-Vercauteren and Brakerski-Gentry-Vaikuntanathan.
10	Implementing Gentry's Fully-Homomorphic Encryption Scheme :Preliminary Report	Implementation report on a variant of Gentry's fully homomorphic encryption scheme (STOC 2009) with some optimizations and bootstrapping functionality. The time to run one bootstrapping operation (on a 1-CPU 64-bit machine with large memory) ranges from 30 seconds for the small "setting to 30 minutes for the large" setting.
11	Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes	This is a fully homomorphic encryption scheme with relatively small key and cipher text size. This scheme follows Gentry by producing a fully homomorphic scheme from a "somewhat" homomorphic scheme. This scheme allows efficient fully homomorphic encryption over any field of characteristic two.
12	Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages	The authors proposed a fully homomorphic encryption scheme, in which the requirement of assumption that somewhat homomorphic encryption has the circular secure, i.e. the scheme can be used to securely encrypt its own secret key is removed. this scheme is based on the ring learning with errors (RLWE) assumption
13	Fully Homomorphic Encryption over the Integers	This is a simple "somewhat homomorphic encryption" method uses only basic modular arithmetic, and use Gentry's method to convert it into a fully homomorphic method. Supports addition and multiplication over the integers.
14	Fully Homomorphic Encryption Using Ideal Lattices	Ideal lattices provide both additive and multiplicative homeomorphisms (modulo a public-key ideal in a polynomial ring that is represented as a lattice), as needed to evaluate general circuits. And it is almost bootstrappable.
15	Fully Homomorphic Encryption without Bootstrapping	A leveled FHE scheme that can evaluate L-level arithmetic circuits with O ($\lambda.L^3$) per-gate computation. It does not use bootstrating technique.

7. CryptDB

The fully homomorphic encryptions have not reached to a position where they can be used directly in critical applications.They are very far in practical usage in terms of efficiency and computational speed and size of cipher text for an average system configuration. In 2011, researchers at MIT funded by two internet power houses, Google and Citigroup, have come up with a practical solution for performing the computations on encrypted data called CryptDB. CryptDB is a scheme which supports realistic and verifiable confidentiality from malicious administrators and malicious programs for applications backed by SQL databases. It works by executing SQL queries over encrypted data using a collection of efficient SQL-aware encryption schemes [5]. The applications considered in this paper are accessing data stored in database provided by the cloud service. Therefore, by using the CryptDB as a database for these applications confidentiality can be achieved from malicious administrators and malicious programs. This solution target two threat models, first, administrator cannot exploit the information of the users. Second, it protects information of unlogged users. The chaining encryption keys to user passwords concept of CryptDB ensures that a data item can be decrypted only by using the password of those users with access to that data. This enables, a database administrator never gets access to decrypted data, and even if all servers are compromised, an adversary cannot decrypt the data of any user who is not logged in [5]. The over head of CryptDB reducing throughput by 14.5 percentages for phpBB, a web forum application, and by 26 percentages for queries from TPCC, compared to unmodified MySQL [5]. In CryptDB data is stored in different onion layers. The main idea is to encrypt every data item into one or more onions layers, that is, each value is dressed in layers of increasingly stronger encryption. Depending on the layer, we can perform certain functionality on the encrypted data items. There are four layers in CryptDB such as Equality, Order, Search, and Addition. Depending on the application requirement the data may be placed in one more layers. The names of layers reflect their functionality in that layer i.e. can perform equality, order preserving, and searching and addition operations on encrypted data with respect to each layer. CryptDB is implemented in C++ and available in open source for Postgre and Mysql databases. The CryptDB mechanism seems very practical and some companies like Google (encrypted Bigquery), Lincoln Laboratory (D4M Accumulo no-SQL engine), MIT (sql.mit.edu) and SAP AG are started using the CryptDB.

8. CONCLUSIONS AND FUTURE SCOPE

One fundamental question is that why should user trust cloud service providers? Is it because there is an agreement between service providers and users? What is the guarantee that service providers adheres to service level agreements? And how often do we read the terms and condition clause in detail before committing online? For example, a wicked service provider's employee or administrator may keep some malicious code or leak information that may result back doors and circumvents the protection mechanism of service provider's environment. The service providers may be using the systems procured from different vendors and there could some situation where these systems are being mismanaged by the inexperienced engineers, this could be a potential loop hole, where this kind of events make attackers an easy task to compromise the system by exploring these loop holes. In IT industry competitors always look for a situation where they try to capture their competitors, in the process they try to explore all possibilities, one possibility could be they may tie up with system suppliers vendors and try to have some bugs in the hardware and this could be exploited. If there are any problems in virtualization and sandboxing mechanisms, further they will be aggravating the service environment. In this paper, we have tried to answer the some of these questions with homomorphic encryption and CryptDB techniques. Applicability of homomorphic encryption and CryptDB are presented to protect data hosted on cloud computing while the applications accessing the data for computational purpose. We have considered the most widely usable social and business applications areas like financial institutes, examination system, mobile communication, e-governance, health industry-mail, e-voting. We have discussed the state of the art partial and fully homomorphic encryption and CryptDB schemes and their applicability to protect information in the cloud computing environment for social and business applications. We have implemented and tested and shown the complete results of four partial homomorphic encryption algorithms over one lakh 10-digit numbers, using Linux virtual machine on VirtualBox,VMPlayer and KVM. The result for four algorithms (namely Paillier, ElGamal, RSA and Benaloh) as performed on the above four different platforms are computed to show their respective overhead values as compared to plain data operations. In case of Paillier Algorithm the overhead is 17, 15, 22 and 12 times for addition operation and 278, 399,518 and 346 times for multiplication operation respectively. Similarly, in case of Elgamal algorithm 1.72, 1.6, 11.7 and 8.9 times for multiplication operation; in case of RSA algorithm 1.79,1.5,3.48 and 1.5 times for multiplication operation and in case of Benaloh algorithm is 5.6, 5.36, 5.48 and 3.5 times for addition operation respectively. These empirical results shows that they are practically feasible and therefore these algorithms can be used in social and business applications and these results definitely helps the service providers to choose the best partial homomorphic algorithm in combination with Virtual Machine. We have also discussed protecting the confidentiality of information in the face of side channel attacks, hardware Trojan horses and zero-day attacks in the service provider's environment. The available solutions of fully homomorphic encryption schemes computational time and cipher text size are very huge. The partial homomorphic encryption mechanism is supported only one operation. Therefore, to use fully homomorphic encryption schemes in our social and business applications hosted on cloud, computational time and cipher text size have to be reasonable to manage them. The available partial homomorphic encryption schemes need to be enhanced to support both addition and multiplication operations.

Disclaimer: The performances of algorithms discussed in this paper are specific to system configuration and key sizes mentioned in the paper. The results may vary with other system configuration and key sizes.

9. REFERENCES

[1] http://www.homelandsecuritynewswire.com /databreaches-Compromise-nearly-8-million-medicalrecords: Data breaches compromise nearly 8 million medical records, published 1 June 2011

[2] http://en.m.wikipedia.org/wiki/PlayStation_Network_outage: Playstation Network outage.

[3] Carlos Aguilar Melchor and Philippe Gaborit, Javier Herranz, Additively Homomorphic Encryption with d-Operand Multiplications.CRYPTO 2010, pp.138-154, 2010.

[4] Ivan Damgard, Mads Jurik: A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System Public Key Cryptography 2001:119-136

[5] Raluca Ada Popa,Catherine M.S.Redfield,Nickolai Zeldovich,and Hari Balakrishnan," CryptDB: Protecting

Confidentiality with Encrypted Query Processing", SOSP'11, October 23-26, 2011, Cascais, Portugal

[6] Yin Hu, A Dissertation on"Improving the Efficiency of Homomorphic Encryption Schemes", May 2013

[7] http://en.wikipedia.org/wiki/sidechannelattack

[8] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2):120126,1978.

[9] T.Okamoto and S. Uchiyama. A New Public-Key Cryptosystem as Secure as Factoring. Eurocrypt 08, LNCS 1403,pp.308-318,1998

[10] http://go.worldbank.org/M1JHE0Z280 (extracted on 18.08.2008)

[11] Brenner, M., Wiebelitz, J., von Voigt, G., Smith, M.: Secret program execution in the cloud applying homomorphic encryption. In: Proceedings of the 5th IEEE International Conference on Digital Ecosystems and Technologies Conference (DEST),pp.114-119.IEEE (2011)

[12] Breuer, P.T., Bowen, J.P.:Typed assembler for a RISC crypto-processor. In:Barthe, G.,Livshits,B.,Scandariato,R.(eds.) ESSoS 2012. LNCS,vol.7159,pp.22-29.Springer, Heidelberg (2012)

[13] Nektarios Georgios Tsoutsos and MichailManiatakos, "Investigating the Application of One Instruction Set Computing for Encrypted Data Computation",in proceeding of SPACE 2013 ,Lecture Notes in Computer ScienceVolume 8204, 2013, pp21-37

[14] Halevi,S.,Shoup, V.:Design and implementation of a homomorphic-encryption library (2012)

[15] Brakerski, Z., Gentry, C., Vaikuntanathan, V.:(Leveled) fully homomorphic encryption without bootstrapping. In:Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, pp. 309-325.ACM(2012)

[16] Ron Rivest, Leonard Adleman,and Michael L.Dertouzos.On data banks and privacyhomomorphisms. In Foundations of Secure Computation,pages 169-180,1978.

[17] Craig Gentry. Fully homomorphic encryption using ideal lattices. In STOC, pages169-178, 2009.

[18] T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms.IEEE Transactions on Information Theory, IT 31(4):469472, July 1985

[19] Marten van Dijk, Craig Gentry, ShaiHalevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In EUROCRYPT, pages 24-43, 2010.

[20] Nigel P.Smart and FrederikVercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Public Key Cryptography-PKC'10, volume6056 of Lecture Notes in Computer Science, pages 420-443.Springer,2010.

[21] Craig Gentry and ShaiHalevi. Implementing gentry's fully-homomorphic encryption scheme. In EUROCRYPT, volume 6632 of Lecture Notes in Computer Science, pages129-148.Springer,2011.

[22] Jean-S ebastienCoron,AvradipMandal,David Naccache, and Mehdi Tibouchi.Fullyhomomorphic encryption over the integers with shorter public keys.In CRYPTO,pages 487-504,2011.

[23] ZvikaBrakerski and VinodVaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages.In CRYPTO, volume 6841, page 501, 2011.

[24] ZvikaBrakerski and VinodVaikuntanathan.Efficient fully homomorphic encryption from (standard) LWE.In FOCS,pages 97-106,2011.References are to full version: http://eprint.iacr.org/2011/344.

[25] Craig Gentry and ShaiHalevi. Fully homomorphic encryption without squashing using Depth-3 arithmetic circuits.In FOCS,pages 107-109,2011.

[26] ZvikaBrakerski, Craig Gentry, and VinodVaikuntanathan. Fully homomorphic encryption without bootstrapping.In Innovations in Theoretical Computer Science (ITCS'12), 2012.Available at http://eprint.iacr.org/2011/277.

[27] Jean-S ÌA¸ ebastienCoron, David Naccache, andMehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers.In Advances in Cryptology-EUROCRYPT 2012, volume 7237 of Lecture Notes in Computer Science, pages 446- 464. Springer, 2012.

[28] Craig Gentry, ShaiHalevi, and Nigel P.Smart.Fully homomorphic encryption with polylog overhead.In EUROCRYPT, pages 465-482,2012.

[29] Craig Gentry, ShaiHalevi, and Nigel P. Smart.Homomorphic evaluation of the aes circuit. In CRYPTO, pages 850-867,2012.

[30] Adriana L opez-Alt, EranTromer, and VinodVaikuntanathan.On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In STOC, pages 1219-1234, 2012.

[31] ZvikaBrakerski.Fully homomorphic encryption without modulus switching from classical gapsvp. In CRYPTO, pages 868-886, 2012.

[32] LinkedIn passwords leaked by hackers:http://www.bbc.co.uk/news/technology-18338956

[33] Matt Bishop and DavidWagner,"Inside Risks",November 2007/Vol.50,No.11 COMMUNICATIONS OF THE ACM

[34] http://en.wikipedia.org/wiki Information_privacy,retrieved 28 Feb 2009.

[35] http//en.wikipedia.org/wiki/Personally_identifiable_information,retrieved 28 Feb 2009.

[36] Google,Inc.User data requests-Google transparency report,Sept.2013. (http: // www.google.com /transparencyreport/userdatarequests/ retrieved 28 Feb 2009.)

[37] "Report of the Defense Science Board Task Force on High Performance Microchip Supply," Defense Science Board, US DoD, Feb. 2005; http://www.acq.osd.mil/dsb/reports/2005-02-HPMS_Report_Final.pdf

[38] J. Lieberman, National Security Aspects of the Global Migration of the U.S. Semiconductor Industry, white paper, Airland Subcommittee, US Senate Armed Services Committee,June Applicability of Homomorphic Encryption and CryptDB in Social and Business Applications 15 2003;Referenceshttp:// lieberman.senate.gov / documents/whitepapers/semiconductor.pdf

[39] S. Adee, ”the Hunt for the Kill Switch,“IEEE Spectrum,vol.45,no.5,2008,pp.34-39.

[40] Innovation at RiskIntellectual Property Challenges and Opportunities,white paper, Semiconductor Equipment and Materials International, June 2008.

[41] C.Gentry,”A fully homomorphic encryption scheme,“ PhD thesis, Stanford University, 2009.

[42] Josh Benaloh,Dense Probabilistic Encryption,SAC 94, pages 120-128, 1994.

[43] S. Goldwasser, S. Micali, Probabilistic Encryption,J.Comp.Sys.Sci.,28,pp.270-299,1984.

[44] https://hcrypt.com/scarab-library/

[45] ”Parents: Cyber Bullying Led to Teen’s Suicide: Megan Meier’s Parents Now Want Measures to Protect Children Online“. ABC News 29 November 2007.

[46] Halderman,J.A.and Schoen,S.D.and Heninger, N. and Clarkson,W.and Paul,W. and Calandrino, J.A. and Feldman, A.J. and Appelbaum,J.and Felten,E.W. LestWe Remember: Cold Boot Attacks on Encryption Keys Proc. 2008 USENIX Security Symposium

[47] AcÄ´siÂÿcmez,O.and Ko.c,C. and Seifert, J.P.Predictingsecret keys via branch prediction Topics in Cryptology- CT-RSA 2007, Springer,2007

[48] Craig Gentry and ShaiHalevi,Implementing Gentry’s fully-homomorphic encryption scheme,Advances in Cryptology-EUROCRYPT 2011,pp.129-148,2011

[49] D. Naccache, J. Stern. A New Public Key Cryptosystem Based on Higher Residues. Proceedings of the 5th ACM CCS, pages 59-66, 1998.

[50] ”Social Network Users Statistics,” http://www.socialnomics.net/2011/08/16/socialnetwork usersstatistics.

[51] http://www.mobilecloudcomputingforum.com

[52] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Advances in Cryptology - Eurocrypt’99, LNCS vol.1592, Springer, 1997, pages 223-238.

[53] Taesoo Kim1, Marcus Peinado2, Gloria Mainar-Ruiz3:1.MIT CSAIL, 2.Microsoft Research,3.Microsoft Research,STEALTHMEM: System-Level Protection Against Cache-Based Side Channel Attacks in the Cloud.

[54] P. Paillier, Trapdooring discrete logarithms on elliptic curves over rings, ASIACRYPT 2000, LNCS 1976,pp.573-584.2000.

[55] A. Kawachi, K.Tanaka,K.Xagawa.Multi-bit cryptosystems based on lattice problems. PKC ’07, pp.315-329.

[56] http://www.verizonenterprise.com/DBIR

[57] SalehAlshomrani and ShahzadQamar,“Cloud Based E-Government:Benefits and Challenges”, INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY SCIENCES AND ENGINEERING, VOL.4,NO.6, JULY 2013.

[58] About Zero Day Exploits (http:// netsecurity. about.com/od/newsandeditorial1/a/aazeroday.htm) Netsecurity.about.com.2010-11-11. Retrieved 2012-01- 08.

[59] J.Tudor Web application vulnerability statistics, June 2013. http://www.contextis.com/files/Web_Application_Vulnerability_Statistics_-_June_2013.pdf.

[60] D.Borelli.The name Edward Snowden should be sending shivers up CEO spines.Forbes, Sept.2

[61] A.Chen.GCreep: Google engineer stalked teens,spied on chats.Gawker,Sept.2010.http:gawker.com/5637234/