Efficient-IBE (Identity based Encryption) based Cloud Data Storage Security Technique for Multi-user Inspection System

Dr. Setu K. Chaturvedi
Professor & Head
Dept. of Comp. Sci. & Engg.
Technocrats Inst.of Tech. Bhopal
Setukchaturvedi@gmail.com

Dr. D.K.Swami
Professor
Dept. of Comp. Sci. & Engg.
V.N.S.I.T, Bhopal
dhirendrakswami@gmail.com

Dr. Gulab Singh
Professor & Head
Dept. of Comp. Sci. & Engg.
L.N.C.T.S., Bhopal
gsinghgulab@gmail.com

***Abstract*— Cloud computing has been envisioned as the next-generation architecture of IT enterprise. In contrast to traditional solutions, where the IT services are under proper physical, logical and personnel controls, cloud computing moves the application software and databases to the large data centers, where the management of the data and services may not be fully trustworthy. This unique attribute, however, poses many new security challenges which have not been well understood. With the development of cloud computing, Data security becomes more and more important in cloud computing. Understanding this trend, the big and giant web based companies like Google Risk coming with opportunity, the problem of data security in Cloud computing become bottleneck of cloud computing. In this paper, we focus on cloud data storage and access security, which has always been an important aspect of quality of service. This paper focuses on information security of cloud computing and data security requirement of cloud services. With the help of efficient Identity Based Encryption (IBE) can be improved the cloud system security. Cloud privacy is maintained with little savings which are available in terms of energy consumption and required channel bandwidth which are also analyzed in this paper.**

Keywords- cloud computing, data security, Encryption, Decryption.

I. INTRODUCTION

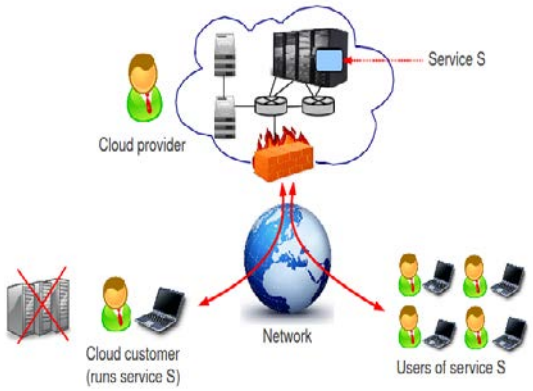
Cloud computing refers to the ability to use software from a Web browser and store the resulting data on the server."Cloud" represents the Internet and cloud computing is really work and software that resides on the Internet or in the "cloud." Although cloud computing is an emerging field of computer science, the idea has been around for a few years.

The Cloud system provides a simple and unified interface between customer and user, allowing customers to focus more on the software itself rather than the underlying framework. Applications on the Cloud include Software as a Service system and Multi-tenant databases. The Cloud system dynamically allocates computational resources in response to customers' resource reservation requests and in accordance with customers predestined quality of service [1, 2]. In this paper we want to set up a security model for cloud computing.

It is the long dreamed vision of computing as a utility, where users can remotely store their data into the cloud so as to enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources. By data outsourcing, users can be relieved from the burden of local data storage and maintenance. Thus, enabling public accessibility for cloud data storage security is of critical importance so that users can resort to an external access party to check the integrity of outsourced data when needed. Specifically, our contribution in this work can be summarized as the following three aspects:

(a) We motivate the public accessing system of data storage security in Cloud Computing and provide a privacy-preserving accessing protocol, i.e., our scheme supports an external use to access user's outsourced data in the cloud without learning knowledge on the data content.

(b) Our scheme is the first to support scalable efficient and secure public accessing in the Cloud Computing. In particular, our scheme achieves batch accessing where multiple delegated accessing tasks from different users can be performed simultaneously by the ATP.



In this paper, we propose EFFICIENT-IBE Identity Based Protocol for securing cloud data with strong authentication mechanism. In this framework cloud data is collected and transmitted through any network, stored at storage site and analyzed on devices like mobile phones, and sent back from devices to the backend servers for subsequent processing and storage.

The rest of the paper is organized as follows: related work is described in section 2, problem domain and proposed solution is described in section 3 and section 4 respectively. Application domain and expected outcome is discussed in section 5 and section 6.

II RELATED WORK

To securely introduce an effective Access of third party (ATP), the following two fundamental requirements have to be met [4]:

(a) ATP should be able to efficiently access the cloud data storage without demanding the local copy of data, and introduce no additional on-line burden to the cloud user;

(b) The third party accessing process should bring in no new vulnerabilities towards user data privacy.

In the paper [1,3] described a formal "proof of irretrievability" (POR) model for ensuring the remote data integrity. Their scheme combines spot-checking and error-correcting code to ensure both possession and irretrievability of files on archive service systems. The author built on this model and constructed a random linear function based homomorphism authenticator which enables unlimited number of queries and requires less communication overhead. In [5] proposed an improved framework for POR protocols that generalizes work. Later in their subsequent work, Bowers et al. [10] extended POR model to distributed systems. However, all these schemes are focusing on static data. The effectiveness of their schemes rests primarily on the preprocessing steps that the user conducts before outsourcing the data file. Any change to the contents of file, even few bits, must propagate through the error-correcting code, thus introducing significant computation and communication complexity. The author has defined the "provable data possession" (PDP) model for ensuring possession of file on entrusted storages [6]. Their scheme utilized public key based homomorphic tags for accessing the data file, thus providing public variability. However, their scheme requires sufficient computation overhead that can be expensive for an entire file. In their subsequent work, the described a PDP scheme that uses only symmetric key cryptography [13]. This method has lower-overhead than their previous scheme and allows for block updates, deletions and appends to the stored file in the secure manner, which has also been supported in our work. However, their scheme focuses on single server scenario and does not address small data corruptions, leaving both the distributed scenario and data error recovery issue unexplored.

III CRYPTOGRAPHIC SECURITY ALGORITHM

It is well-known that Cloud networks have been very popular in the research community for the past years. Cloud motes have had many restrictions which pose great barriers and challenges for the Cloud network capabilities. Power consumption, energy, storage space restrictions, computational weakness and transmission inaccuracies are some examples of the restrictions in Cloud networks.

The research community has been widely investigating Elliptic Curve Cryptography (ECC) and especially Bilinear Pairing. Many schemes (some of them very successful) have been developed that can provide encryption, digital signatures, threshold decryption or signatures, key exchange, identity-based encryption and many more. These can be used as proof of concept of the different tools bilinear pairing can provide for the security field. In this paper we have used the elliptic curve cryptography in the efficient manner [7].

Elliptic Curve Cryptography (ECC) has been a very interesting research field for many applications especially in Cloud networks. The main reason for this is the restrictions on resources posed by Cloud server [8]. ECC alleviates these restrictions by using small prime fields. At the same time, the evolution of hardware technology has built new, sophisticated motes with more capabilities which open new ways for Cloud network applications. The main reason is the small key sizes it uses that can lead to efficient cryptographic calculations [12, 14].

Elliptic curve cryptography [9] has a history of almost a hundred years. It has its origins in mathematics and number theory. Though, by considering the discrete logarithm problem and mapping it to discrete logarithms on groups of points of an elliptic curve we can realize the alternative use of elliptic curves in elliptic curve cryptography. In the mid-nineties, researchers proposed using elliptic curves for public-key cryptosystems. Since then, elliptic curve cryptography has been widely studied and there are a significant number of systems and protocols where it is being used. Many of these systems are of commercial acceptance and this is only the beginning as the research community is constantly accepting and extending the capabilities and uses of elliptic curves.

a. Identity Based Encryption (IBE)

Our solutions are based on a type of cryptographic primitive known as identity-based encryption (IBE) [11, 15]. After an initial setup phase, IBE allows a public key to be generated from an arbitrary string. The corresponding secret key can be derived separately by a trusted party. For example, Al- ice may want to encrypt a message for the user in charge on Monday. Alice can independently generated a public key using the strings "Monday" and "user" to encrypt her data without further contact with the trusted party. To decrypt the message, a user will have to convince the trusted party that he is a user in charge on Monday. The same user working on Tuesday cannot decrypt messages encrypted using the string "Tuesday" and "user" even if he knows the secret key from Monday.

The simple example cannot be easily accomplished with- out using IBE. Alice can try to generate many public/secret key pairs, one for each occasion. However, Alice will have to store the secret key created with the trusted party each time a new public/secret key pair is generated. Otherwise, the trusted party cannot derive the secret key on its own. This is inefficient. Another possible alternative is for Alice to include some instructions with each of her message, and encrypt everything with the trusted parties public key. When a user receives an encrypted message, the user will forward it to the trusted party. The trusted party can decrypt can obtain Alice's instructions. The trusted party will release the message to the user only if he meets Alice's instructions. This solution is also inefficient since a NETWORK may generate many pieces of data, each of which has to be forwarded to the trusted party for decryption. Using IBE, the user only needs to be given a single secret key once.

IV. PROBLEM DOMAIN

From the perspective of data security, which has always been an important aspect of quality of service, Cloud Computing inevitably poses new challenging security threats for number of reasons.

Firstly, traditional cryptographic primitives for the purpose of data security protection cannot be directly adopted due to the users' loss control of data under Cloud Computing. Therefore, verification of correct data storage in the cloud must be conducted without explicit knowledge of the whole data. Considering various kinds of data for each user stored in the cloud and the demand of long term continuous assurance of their data safety, the problem of verifying correctness of data storage in the cloud becomes even more challenging.

Secondly, Cloud Computing is not just a third party data warehouse. The data stored in the cloud may be frequently updated by the users, including insertion, deletion, modification, appending, reordering, etc. To ensure storage correctness and integrity authorized user will be allowed to access/modify the data.

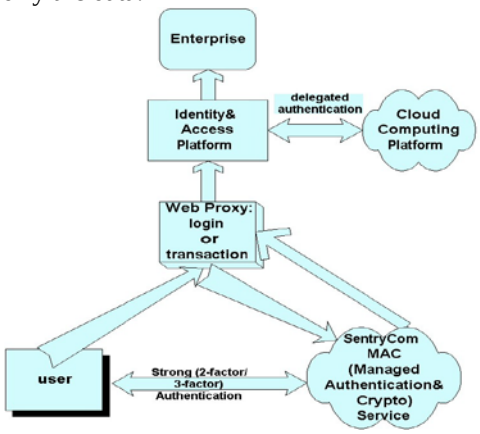


Figure2. System Architecture for Cloud

To enable privacy-preserving public accessing for cloud data storage under the aforementioned model, our protocol design should achieve the following security and performance guarantee in Figure 2:

(i) Public accessibility: to allow ATP to verify the correctness of the cloud data on demand without retrieving a copy of the whole data or introducing additional on-line burden to the cloud users.

(ii) Storage correctness: to ensure that there exists no cheating cloud server that can pass the access from ATP without indeed storing users' data intact.

(iii) Privacy-preserving: to ensure that there exists no way for ATP to derive users' data content from the information collected during the accessing process.
(iv) Batch accessing: to enable ATP with secure and efficient accessing capability to cope with multiple accessing delegations from possibly large number of different users simultaneously.
(v) Lightweight: to allow ATP to perform accessing with minimum communication and computation overhead.

V. Proposed Solution

In this paper, we utilize the protocols based on identity-based encryption (IBE) to achieve a privacy-preserving public accessing system for cloud data storage security while keeping all above requirements in mind, while allowing flexible access to stored data. To support efficient Handling of multiple accessing tasks, we further explore the technique of bilinear aggregate signature to extend our main result into a multi-user setting, where ATP can perform multiple accessing tasks simultaneously. Extensive security and performance analysis shows the proposed schemes are provably secure and highly efficient. We also show how to extent our main scheme to support batch accessing for ATP upon delegations from multi-users.

In this paper, we propose Efficient-IBE scheme which is suitable for Cloud storage security. We implement a proof-of-concept of our schemes based on Efficient-IBE on commercially available Cloud storage similar to the ones used in a NETWORK; While IBE schemes have been suggested by previous researchers to protect data. The next section presents our protocols in Figure 3.

Algorithm:

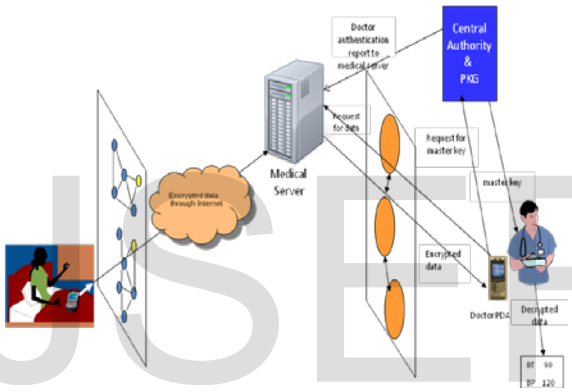


Figure 3: General architecture of proposed solution.

It has four main components (1) NETWORK, which are on patient's body (2) Cloud server or storage (3) Central Authority (CA) and (4) Service provider assistant SPA. Data having values of physiological parameter is generated by NETWORK, which are then encrypted and sent to the storage site using Internet. Storage site collects and stores the encrypted data. When this data is needed by the Service provider, a request is sent first to the central authority for master key generation. For that CA checks the authentication and if authentication is valid, it generates the master key and sends it to the user's SPA. User then request for data by his SPA to the medical server through internet. Now the storage site sends data to user SPA. This data is decrypted by user's SPA using the master key.

Efficient-IBE shares two properties with conventional IBE, namely the ability to use an arbitrary string to generate a public key, and the ability to generate a public key separately from the corresponding secret key. We begin by first reviewing Elliptic Curve Cryptography (ECC), a public key primitive suitable for NETWORK [13], followed by the modifications made to derive Efficient-IBE. Finally, we present our protocols based on Efficient-IBE.

Basic ECC Primitives

To setup ECC, we first select a particular elliptic curve E over GF(p), where p is a big prime number. We also denote P as the base point of E and q as the order of P, where q is also a big prime. We then pick a secret key x, and the corresponding public key y, where $y = x \cdot P$, and a cryptographic hash function h(). Finally, we have the secret key x and public parameters (y, P, p, q, h (.)).

We denote encrypting a message m using public key y as EccEncrypt(m, y). The resulting ciphertext is denoted by c. The decryption of ciphertext c using the secret key x is given as EccDecrypt(c, x). The algorithms for EccEncrypt and EccDecrypt are found in Algorithm 1 and Algorithm 2 respectively.

Algorithm1: EccEncrypt(m, y)

1: Generate a random number r ∈ GF(p). Encrypt m with r, $E_r(m)$
2: Calculate $A_r = h(r) \cdot y$
3: Calculate $B_r = h(r) \cdot P$
4: Calculate $\alpha_r = r \oplus x(Ar)$, where x(Ar) is the x coordinate of A_r
5: Return ciphertext c = $< h \alpha_r, B_r, E_r(m)>$

Algorithm2: EccDecrypt(c, x)

1: Calculate $x \cdot B_r = x \cdot h(r) \cdot P = h(r) \cdot y = A_r$
2: Determine the x coordinate, x(Ar)
3: Derive symmetric key r with $\alpha_r \oplus x(A_r) = r \oplus x(Ar) \oplus x(Ar) = r$
4: Apply r to Er(m) to return m

a. SYSTEM DOMAIN

For the implementation purpose we need a system having Windows XP operating system with JAVA, Swing, RMI, J2ME (Wireless Toolkit) software.

b. Application Domain Modules:

Privacy-Preserving Public Accessing Module: Homomorphism authenticators are unforgivable verification metadata generated from individual data blocks, which can be securely aggregated in such a way to assure an accessor that a linear combination of data blocks is correctly computed by verifying only the aggregated authenticator. Overview to achieve privacy-preserving public accessing, we propose to uniquely integrate the homomorphic authenticator. In our protocol, the linear combination of sampled blocks in the server's response is masked with randomness generated by a pseudo random function (PRF).

The proposed scheme is as follows: From the basic ECC primitives, we derive the following Efficient-IBE primitives:

- Setup,
- Keygen,
- Encrypt, and
- Decrypt.

For ease of explanation, we assume that all primitives are executed by the patient. The actual protocols involving the patient, CA and user are explained in the next section of protocol.

The intuition behind is to let a Cloud independently generate a public key on-the-fly using an arbitrary string. For example, a collecting EKG readings on Monday 1 pm will first create a string str = (monday|1 pm|EKG). Using this string, the Cloud can derive a public key, ystr to encrypt the data and send it to the storage site. There is no corresponding secret key created. In fact, the Cloud cannot create the secret key needed to decrypt the message.

When the patient wishes to release this information to a user, the patient can derive the corresponding secret key, xstr, by using the same string str = (monday|1pm|EKG). This secret key only allows the user to decrypt messages encrypted by a Cloud using the same string. This simplifies the key management, since the patient can generate the secret key on-demand without keeping track of which keys were used to encrypt which data. The only requirement is that the string used to describe the event is the same.

Setup: We select an elliptic curve E over GF(p), where p is a big prime number. We also denote P as the base point of E and q as the order of P, where q is also a big prime. A set of n secret keys $x1, \cdots, xn \in GF(q)$ is chosen to generate the master secret key,

$$X = (x1, \cdots, xn).$$

The n public keys are then generated to make up the master public key,

$$Y = (y1, \cdots, yn)$$

where $yi = xi \cdot P$, $1 \leq i < n$. Finally, a collision resistant one-way hash function $h: \{0, 1\} \rightarrow \{0, 1\}^n$ is chosen. The parameters (y, P, p, q, h(.))are released as the system public parameters.

Keygen: To derive a secret key xstr corresponding to a public key generated by a string str, the patient executes Keygen(str) = xstr,

$$xstr = \sum_{i=1}^{n} hi\ (str) \cdot x_i$$

where h_i(str) is the i-th bit of h(str).

Encrypt:

To encrypt a message m using a public key derived from string str, the Cloud does Encrypt(m, str) to determine the ciphertext c. Alg. 3 shows the process. Note that Alg. 3 lines 1 and 2 need only be run once to derive the public key ystr.

Algorithm 3 Encrypt (m, str)

1: Determine string str using agreed upon syntax
2: Generate public key y_{str} where

$$y_{str} = \sum_{i=1}^{n} hi\ (str) \cdot y_i$$

3: Execute EccEncrypt(m, ystr) to obtain c.

Decrypt:

The user executes EccDecrypt(c, xstr) to obtain the original message m which was encrypted using a secret key derived from str. The process is shown in Alg. 4.

Algorithm 4: Decrypt(c, str)

After authorization and Key generation

1. User executes **EccDecrypt(c, x_{str})** to obtain m

c. *NETWORK SECURITY PROTOCOLS*

We first describe the initialization phase where the patient configures the NETWORK, followed by the data collection phase which outlines how a Cloud encrypts the collected data. The data transfer phase describes how a NETWORK transfers data to a storage site, and finally, the query phase which occurs when a user needs to obtain data from the storage site.

We assume that an agreed upon syntax is used to describe the string needed to derive a public key, and this description is termed as str. For example, the patient deciding to collect data on a hourly basis will set the Clouds in the NETWORK to affix a timestamp rounded to the nearest hour when creating str. In other words, two EKG readings collected on Monday at 1:05 pm and 1:20 pm will both be described using the same string str = {monday|1 pm|EKG}.

As mentioned earlier, we assume an honest-but-curious storage site which will try to learn the contents of the stored data, but will otherwise not delete the stored data. We also assume a separate security mechanism is in place so that only the patient can store NETWORK data onto the storage site.

Initialization: The patient first executes Setup to obtain the master secret key X = (x1,..,·xn), and public parameters (y, P, p, q, h(.)). The patient loads the parameters (y, P, p, q, h(.)) into every Cloud in the NETWORK. The patient then registers the master secret key together with additional instructions with the CA.

Data collection: Let the Cloud collect data d at event str with the flag that is the given bit string. The Cloud executes Alg. 5 to encrypt its data.

Algorithm 5: Cloud encrypting data

1: Derive the string str.
2: Generate Flag, where flag is a known bit-string
3: Calculate c =Encrypt(str,d)
4. send (Flag, c1) to storage site.
tuple (Flag, c1) is then stored in Cloud memory.
The flag is a commonly known bit-string several bits long.

Data transfer: Periodically, each Cloud in the NETWORK will transfer its data to the storage site. This is done by first aggregating all the data into a cell-phone like device that may be either PDA or it may be any built-in device within the body Cloud network that is equipped with GSM service [2]. The cell-phone then forwards the aggregated data to the storage site. Assuming that there are k tuples generated by the NETWORK, the cell-phone will forward the set $\{(c^1, flag^1), \cdots, (c^k, flag^k)\}$. Alternatively, a Cloud with enough storage capacity can opt to store the data within the Cloud itself. In this case, there is no data transfer process.

Querying: A user wishing to obtain data collected under some str will first contact the CA for permission. After the CA agrees, the CA will run Keygen(str) to derive the corresponding secret key X_{str} needed to decrypt data. The user then contacts the storage site and retrieves the data as shown in Alg. 6. When the data is stored within the Cloud, the role of the storage site will be executed by the Cloud.

Algorithm 6: User querying for data

1. User send certificate (User-Id) to CA and PKG (KMS) and Patient-Id
2. with known string for Authentication and key generation.
3. CA will check the authorization and Access Permission of User
4. If "Authenticated" then
   4.1. CA runs **Keygen(str)** to derive x_{str}
   4.2. CA sends the Private key (Xstr) to user
5. User send Patient-id and flag string to Storage Site. And then
   5.1. For every $(c^i, flag^i)$ $i \in K$ for patient do
   5.2. Storate site mathes $flag^i$ string with flag string given by user
   5.3. If flag string = = $flag^i$ then
   5.4. Storage site sends corresponding encrypted c^i to user
   5.5. User execute the **EccDecrypt (c^i , str)**
   5.6. User accept the patient data
   5.7. End if
   5.8. End for

Since all the data is encrypted, the storage site checks the flag of each tuple in database and if the flag matches then storage site return a specific encrypted tuple to the user , instead, the storage site simply lets the user try to decrypt each tuple (c^1) belonging to the patient. The reason for returning specific c^i to the user instead returning all tuples is to improve efficiency of channel by reducing the traffic on the channel and hence it will save bandwidth of the channel.

Batch Accessing Module:

With the establishment of privacy-preserving public accessing in Cloud Computing, ATP may concurrently handle multiple accessing delegations upon different users' requests. The individual accessing of these tasks for ATP can be tedious and very inefficient. Batch accessing not only allows ATP to perform the multiple accessing tasks simultaneously, but also greatly reduces the computation cost on the ATP side.

Data Dynamics Module:

Hence, supporting data dynamics for privacy-preserving public risk accessing is also of paramount importance. Now we show how our main scheme can be adapted to build upon the existing work to support data dynamics, including block level operations of modification, deletion and insertion. We can adopt this technique in our design to achieve privacy-preserving public risk accessing with support of data dynamics.

Cloud application services or "Software as a Service (SaaS)" deliver software as a service over the Internet, eliminating the need to install and run the application on the customer's own computers and simplifying maintenance and support.
Cloud platform services, also known as platform as a service (PaaS), deliver a computing platform and or solution stack as a service, often consuming cloud infrastructure and sustaining cloud applications. It facilitates deployment of applications without the cost and complexity of buying and managing the underlying hardware and software layers.
Cloud infrastructure services, also known as "infrastructure as a service" (IaaS), deliver computer infrastructure – typically a platform virtualization environment – as a service, along with raw (block) storage and networking. Rather than purchasing servers, software, data-center space or network equipment, clients instead buy those resources as a fully outsourced service. Suppliers typically bill such services on a utility computing basis; the amount of resources consumed (and therefore the cost) will typically reflect the level of activity.

VII. EXPERIMENTAL ANALYSIS

a. *Analysis of Basic Primitives*

Our Setup is similar to that of the basic ECC setup scheme, except that instead of picking a single secret x, our Setup picks n secrets and n corresponding public keys. Knowing only one xstr and h(str), the user cannot determine the patient's master secret X since there are n unknown xi. The user is only able to determine X when he has in this possession n different secret keys x1 str, · · · , xn str. The use of xstr and ystr as the private key and public key derived from string str does not violate the discrete logarithm property of ECC where, given a y = x · P, it is infeasible to determine x given y and P, since both are simply the result of addition of points. Also, both Encrypt and Decrypt are secure since both rely on well established ECC encryption and decryption methods.

b. *Analysis of Proposed Security Protocols*

Our protocols protect the privacy of the patient's data by encrypting all the information before forwarding the data to the storage site. After a Cloud collects the data, the Cloud encrypts the data using Encrypt, resulting in the tuples (c1). The storage site receives an aggregated set of tuples $\{(c^1, flag^1), \cdots, (c^k, flag^k)\}$. Since all tuples are in cipher text, the storage site learns nothing about the patient's data.

The protocols also prevent unauthorized access to the patient's data. Each piece of data collected by a Cloud is encrypted with a ystr, the public key derived from the string str. When the user receives permissions to access data encrypted under str, the user receives the secret key Xstr, which cannot be used to decrypt any other cipher text not encrypted using ystr.

A compromised Cloud does not allow the adversary to obtain any useful data about a patient from the storage site since the Cloud only stores the publicly known parameters. At most the adversary obtains the cipher text pair (c1, flag). The adversary can try to launch a matching attack by first creating many public keys using different strings str. The adversary then encrypts all possible values using the different public keys to determine whether there is a match for the tuples (c1, flag). This is possible since the number of potential EKG readings for example is bounded. However, c1 contains a random number n generated by the Cloud. Since the adversary cannot predict the value of n, the matching attack fails.

Finally, our protocols provide flexibility. The string str can be used to specify access to the data, with using certificates. For instance, consider the string str = {Date | ER | User} used to encrypt data. A user wanting to obtain the corresponding secret key will have to convince the CA that he is indeed an ER user on the given date. The process of specifying what str to construct can be programmed by the patient without additional permissions from the CA.

VIII. CONCLUSION

For the correctness and safely of data encryption/decryption algorithm is used. The persons having valid key will be allowed to access data from cloud. We will simulate an environment where valid user will store data in cloud. Encryption will be done to ensure safely and mishandled/misread by unauthorized persons. The person who has valid decryption key will only able to see/access the data. This paper has presented the design of a system of compact, wearable, wireless body sensing devices implanted in the human body. The novel achievement in this paper that we have proposed is the improvement in the existing protocol for data encryption, decryption and transfer between NETWORK, storage site and user with the need for high data rates.

In this proposal some saving of encryption and decryption, required bandwidth of channel and energy consumption of Cloud can be achieved. In this paper, IBE has been used with Elliptic Curve Cryptography (ECC) which makes strong public key cryptography system for the purpose of data encryption and decryption. In summary, the goal of this paper was to look deeper into network application, considering their dependability and safety, trying to make this application more safe and reliable to use.

REFERENCES

[1] Dai Yuefa, Wu Bo, Gu Yaqiang, Zhang Quan, Tang Chaojing, Data Security Model for Cloud Computing, ISBN 978-952-5726-06-0 Proceedings of the 2009 International Workshop on Information Security and Application (IWISA 2009) Qingdao, China, November 21-22, 2009.
[2] Cong Wang, Qian Wang, and Kui Ren Wenjing Lou "Ensuring Data Storage Security in Cloud Computing" Proc. of ICDCS '08, pp. 411–420,2009.
[3] A. Juels and J. Burton S. Kaliski, "PORs: Proofs of Retrievability for Large Files," Proc. of CCS '07, pp. 584–597, 2007.
[4] H. Shacham and B. Waters, "Compact Proofs of Retrievability," Proc. of Asiacrypt '08, Dec. 2008.
[5] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability: Theory and implementation," Cryptology ePrint Archive, Report 2008/175, 2008, http://eprint.iacr.org/.
[6] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. Of CCS '07, pp. 598–609, 2007.
[7] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. of SecureComm '08, pp. 1, 2008.
[8] T. S. J. Schwarz and E. L. Miller, "Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage," Proc. of ICDCS '06, pp. 12–12, 2006.
[9] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard, "A Cooperative Internet Backup Scheme," Proc. of the 2003 USENIX Annual Technical Conference (General Track), pp. 29–41, 2003.
[10] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," Cryptology ePrint Archive, Report 2008/489, 2008, http://eprint.iacr.org/.
[11] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple- Replica Provable Data Possession," Proc. of ICDCS '08, pp. 411–420,2011.
[12] D. L. G. Filho and P. S. L. M. Barreto, "Demonstrating Data Possession and Uncheatable Data Transfer," Cryptology ePrint Archive, Report 2006/150, 2006, http://eprint.iacr.org/.
[13] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Accessing to Keep Online Storage Services Honest," Proc. 11th USENIX Workshop on Hot Topics in Operating Systems (HOTOS '07), pp. 1–6, 2007.
[14] Darrel Hankerson, Alfred Menezes, and Scott Vanstone. Guide to Elliptic Curve Cryptography. Springer-Verlag, fourth edition, 2004.
[15] Tan, H Wang, S Zhong, Q Li " Body sensor network security: an identity-based cryptography approach, Proceedings of the first ACM conference on Wireless network, 2008 - portal.acm.org.