

**2. SOFTWARE REQUIREMENT ANALYSIS AND SPECIFICATION**

**2.1. RELATED WORK**

With the routing path of each packet, many measurement and diagnostic approaches are able to conduct effective management and protocol optimizations for deployed WSNs consisting of a large number of unattended sensor nodes. For example, PAD depends on the routing path information to build a Bayesian network for inferring the root causes of abnormal phenomena.

Path information is also important for a network manager to effectively manage a sensor network. For example, given the per-packet path information, a network manager can easily find out the nodes with a lot of packets forwarded by them, i.e., network hop spots. Then, the manager can take actions to deal with that problem, such as deploying more nodes to that area and modifying the routing layer protocols.

Furthermore, per-packet path information is essential to monitor the fine-grained per-link metrics. For example, most existing delay and loss measurement approaches assume that the routing topology is given as a priori.

The time-varying routing topology can be effectively obtained by per-packet routing path, significantly improving the values of existing WSN delay and loss tomography approaches.

**2.1.1. Literature Survey**

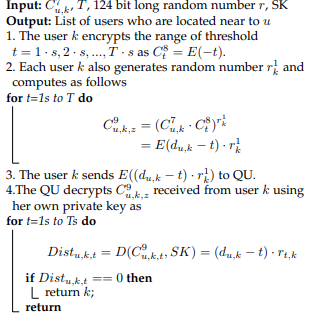
In the paper titled "I Path: Path Inference in Wireless Sensor Networks," authored by Yi Gao, Wei Dong, Chun Chen, Jiajun Bu, Wenbin Wu, and Xue Liu, the authors propose a method for inferring paths in wireless sensor networks. This research, published in the IEEE/ACM Transactions on Networking, Volume 24, Issue 1, in February 2016, addresses the challenge of accurately determining paths within wireless sensor networks.

Data protection is outsourced. In safe cloud-based information, cryptographic encryption techniques are typically employed. Traditional public-key encryption approaches are utilised for outsourcing user-focused access control. IDE is a promising solution for removing reliable certificates from users. Wei et al. employed IBE for data sharing on mobile computers. He and others used IBE to build a handshake system for patient safety. Broadcast identity encryption (IBBE) enhances the IBE to accommodate multi-receiver encryption users while encrypting a message. Deng et al. used IBBE to allow many authorised viewers to access the same outsourced file for the cloud system. A number of revokable schemes should be revoked by some IBBE recipients. Inter-domain transformation. Blaze et al. first used proxy encryption to manage their ciphertext transformation encryption systems. The PRE allows users to translate ciphertext from Alice into chip text under the public Bob password. Ateniese et al. defined PRE as interactive and non-interactive in different categories, including PRE, Single Store and Multi Hop. Twin-way PRE. Much was done to improve PRE's safety and efficiency, in particular unidirectional PRE. Libert and Vergnaud have constructed the first one-way system. Cao et al. suggested an independent PRE path to allow users to adopt a preference external

visitor path. Guo et al. introduced PRE proxy accounting unidirectional encryption key. Green and Ateniese offered the first PRE (IBPRE) ID, which would extend the PRE to PRE ID and IBE ID. Tseng and Chu introduced short code text IBPRE and decryption keys that can put confidential data holders at risk, i.e. the proxy server coalition and user access. This problem has been overcome by the revoked IBPRE cloud system Liang et al. The interaction between data proprietors and the processing authority of key generators is vital for the efficiency of this system. Xu et al. stated that IBBE should be based on IBBE when integrated into a PRE system. Additional PRE expansions like PREs, etc. are feasible except for IBPRE. These PRE systems primarily transform ciphertexts into the same encryption system so that ciphertexts are converted into a different format. Cross-domain transformation. There are numerous transformation plans for cross-domain encryption. Matsuo has connected the normal public key encryption system and identity-base encryption system to convert public ciphertext systems into IBE ciphertext. Mizuno and Doi additionally offer a one-way PRE system to process and save the user chip encoding system. Jiang et al. presented a domain-cross encoding system which would recently connect

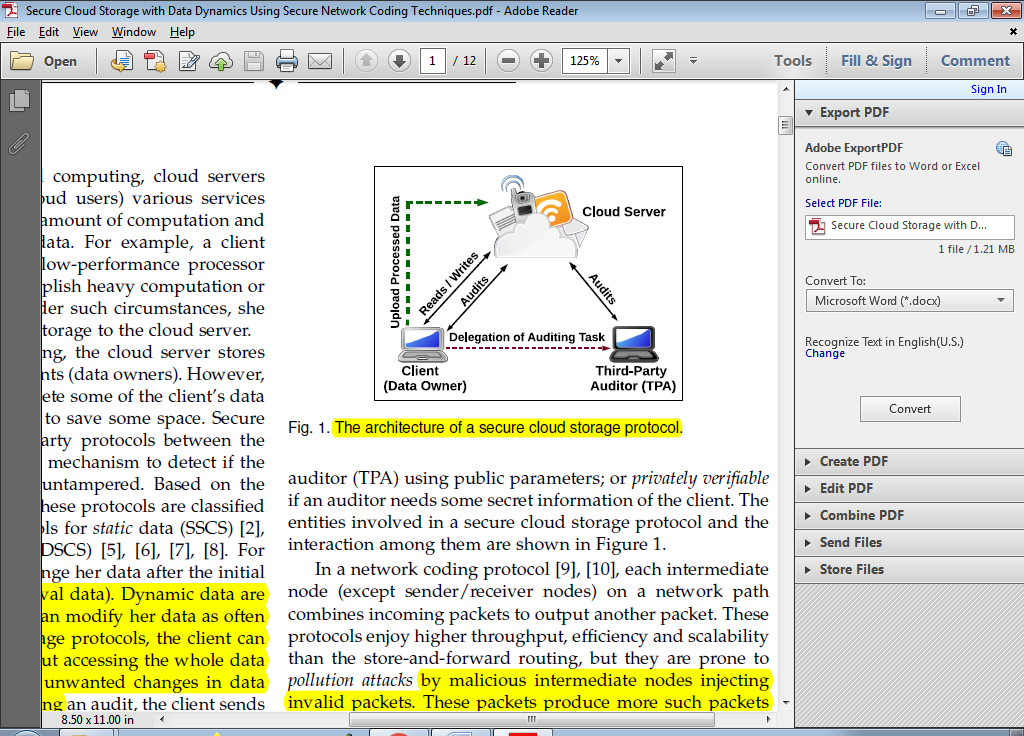
traditional public key encoding and identification to each public key user. This document aims at handling the cross-domain and so saves expenses for the administration of certificate identification settings. This page includes IBE transfers to IBBE so that, even when encrypted, information is shared with other users Outsourced data protection. Cryptographic encryption methods have been extensively used to secure data outsourced to clouds. Traditional public-key encryption methods are applied to achieve user-centric access control on outsourced data [4], [5]. Identity-based encryption (IBE) [6] is a promising cryptographic tool which eliminates trusted certificates for all users. Wei et al. [7] exploited IBE to secure data sharing in mobile computing environments. He et al. [8] employed IBE to construct a handshake scheme in healthcare social network to secure data exchanged in patients. Identity-based broadcast encryption (IBBE) [9] extends IBE to support multi-receiver encryption in the sense that a user encrypts a message once for multiple intended receivers. In light of such useful feature, Deng et al. [10] utilized IBBE in cloud storage systems to allow multiple authorized visitors to access the same outsourced file. To revoke some recipients from the initial receiver set of the IBBE ciphertext, a number of revocable IBBE schemes are proposed [11], [12], [13], [14]. Inter-domain Transformation. Blaze et al. [15] first introduced the concept of proxy re-encryption to handle ciphertext transformation within an encryption system. With this PRE, a user can transform a ciphertext generated under Alice’s public key into a ciphertext under Bob’s public key. Ateniese et al. [16] classified PRE into different categories: bidirectional and unidirectional PRE, single-hop and multihop PRE, interactive and non-interactive PRE. Many efforts have been made to improve efficiency and security of PRE and most of them focus on unidirectional PRE. Libert and Vergnaud [17] presented the first unidirectional PRE scheme. Cao et al. [18] proposed the autonomous path PRE scheme to enable a user to designate a path of preferred authorized visitors to his outsourced data. Guo et al. [19] introduced accountability into unidirectional PRE to identify the proxy which abuses its re-encryption keys. By combining PRE and IBE, Green and Ateniese [20] proposed the first identity-based PRE (IBPRE), which is an extension of PRE in identity-based settings. Chu and Tzeng [21] presented an IBPRE scheme with short ciphertexts and decryption keys, while it is vulnerable to collusion attack, i.e., the coalition of the proxy server and the authorized users could compromise the secret information about data owners. Liang et al. [22] overcome this security issue by proposing the cloud-based revocable IBPRE scheme. This scheme requires the interaction between data owners and a key generator authority for each transformation, which may result an efficiency problem. Xu et al. [23] proposed an IBBE-based PRE scheme by introducing IBBE into PRE. Apart from IBPRE, there are other extensions of PRE, such as attribute-based PRE [24], [25], time-based PRE [26], function-based PRE [27], etc. However, these PRE schemes mainly provides ciphertext transformation in the same encryption system, that is, ciphertexts cannot be converted into another format. Cross-domain transformation. There are a few schemes achieving cross-domain encryption transformation. Matsuo [28] linked the traditional public-key encryption and identity-based encryption by allowing to transform a ciphertext of public key systems into a ciphertext of IBE systems. Mizuno and Doi [29] also proposed a unidirectional PRE scheme that transforms ciphertexts of an attributebased encryption system into ciphertexts of an IBE system, while requiring users to interact with each other and store additional information for transformation. Recently, Jiang et al. [30] proposed a cross-domain encryption switching scheme that connects traditional public-key encryption and identity-based encryption, while it requires cryptographic certificates for all the users in the public-key encryption system. This paper aims at addressing cross-domain transformation in identity-based settings; thus saves the cost for certificate management. Moreover, this paper provides encryption transformation from (one-receiver) IBE system to (multi-receiver) IBBE system so that one’s data can be shared with more users even though the data have already been encrypted

**2.1.2 Existing algorithms/ Techniques**



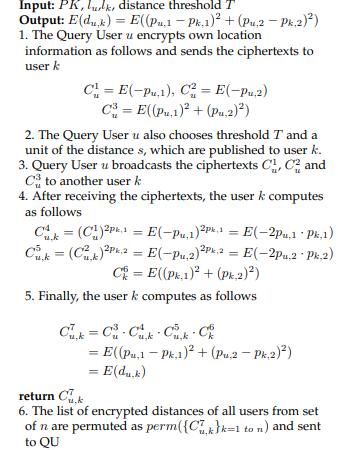
**2.2. RESEARCH METHODOLOGY**

**2.2.1. System Architecture**



**Fig.2.2.1: System Architecture**

**2.2.2 Proposed Algorithms / Techniques**



**2.3 PROPOSED MODULES**

**Data Owner**

In this module, the data owner uploads their data with its chunks in the cloud server. For the security purpose the data owner encrypts the data file and then store in the cloud. The Data owner can have capable of manipulating the encrypted data file. And the data owner can set the access privilege to the encrypted data file and performs the following operations Register with service period, Login, Browse, encrypt, Upload with mac, Verify file, Delete file, View Cloud Details, Find Trustworthiness of cloud, Send Review about cloud usage, Recommend the cloud.

**Cloud Servers**

The cloud service provider manages a cloud to provide data storage service. Data owners encrypt their data files and store them in the cloud for sharing with data consumers. To access the shared data files, data consumers download encrypted data files of their interest from the cloud and then decrypt them.

**Third Party Arbitrator**

Third party auditor (TPA), who has capabilities to manage or monitor the outsourced data under the delegation of data owner, who has expertise and capabilities that a common user does not have, for periodically auditing the outsourced data. This audit service is significantly important for digital forensics and credibility in clouds.

**3.4.4. Data Consumer (End User)**

The Cloud User who has a large amount of data to be stored is integrated and downloaded.

**2.4. USER CONSTRAINTS**

User Constraints for project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL CONSTRAINTS
* TECHNICAL CONSTRAINTS
* SOCIAL CONSTRAINTS

**ECONOMICAL CONSTRAINTS**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### **TECHNICAL CONSTRAINTS**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL CONSTRAINTS**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**2.5. HARDWARE REQUIREMENTS**

**HARDWARE REQUIREMENTS**

Processor : I3 or higher

Speed : 2.9 GHz

RAM : 4 GB (min)

Hard Disk : 160 GB

**2.6. SOFTWARE REQUIREMENTS**

* **Operating system :** Windows 7 Ultimate
* **Coding Language :** Python
* **Back-End :** Django-ORM
* **Designing :** Html, CSS, JavaScript
* **Data Base :** MySQL (WAMP Server)

**2.7. NON-FUNCTIONAL REQUIREMENTS**

Non-functional requirements are the constraints that must be adhered during development. They limit what resources can be used and set bounds on aspects of the software’s quality.

**User Interfaces**

The User Interface is a GUI developed using Python.

**Software Interfaces**

The main processing is done in Java and console application.

**Manpower Requirements**

5 members can complete the project in 2 – 4 months if they work fulltime on it.

### **SDLC METHODOLOGIES**

SDLC stands for Software Development Life Cycle. A Software Development Life Cycle is essentially a series of steps, or phases, that provide a model for the development and lifecycle management of an application or piece of software. The methodology within the SDLC process can vary across industries and organizations, but standards such as ISO/IEC 12207 represent processes that establish a lifecycle for software, and provide a mode for the development, acquisition, and configuration of software systems.

SDLC consists of following phases

* Requirements Gathering Stage.
* Analysis Stage.
* Designing Stage.
* Development Stage.
* Testing Stage.
* Maintenance Stage.

**Requirements Gathering Stage**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions are termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



**Fig 2.7.1: Requirements Gathering Stage**

These requirements are fully described in the primary deliverables for this stage, the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages. The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

**Analysis Stage**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



**Fig 2.7.2: Analysis Stage**

The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high-level estimates of effort for the out stages.

**Designing Stage**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are in ended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.



**Fig 2.7.3: Designing Stage**

When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development Stage**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artefacts will be produced. Software artefacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artefacts, and an online help system will be developed to guide users in their interactions with the software.

The RTM will be updated to show that each developed artefact is linked to a specific design element, and that each developed artefact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.



**Fig 2.7.4: Development Stage**

The design document is set to development stage and then further processing happens like splitting the data for software, online help, and project schedule, implantation, test plan and updating the matrix.

**Testing**

During the integration and test stage, the software artefacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data and production user list are compiled into the Production Initiation Plan. The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

**Maintenance**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category.

**WATERFALL MODEL**

The Waterfall Model is a direct progressive stream. In which progress is viewed as streaming dependably downwards (like a wellspring) through the hours of programming execution. This infers that any stage in the improvement cycle begins given that the previous stage is done. The fountain approach doesn't portray the collaboration to get back to the past stage to manage changes in need. The outpouring approach is the earliest philosophy and most all around understood that was used for programming improvement.

Waterfall model is broken down into multiple phases:



**Fig 2.7.3.1: Water fall model**

**Advantages**

* Simple to clarify for the clients.
* Structures approach.
* Stages and exercises are distinct.
* Assists with arranging and timetable the task.