إقـــــرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

A Model to Ensure the Integrity of User's Data in the Cloud

أقر بأن ما اشتملت عليه هذه الرسالة إنما هو نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه حيثما ورد، وإن هذه الرسالة ككل أو أي جزء منها لم يقدم من قبل لنيل درجة أو لقب علمي أو بحثي لدى أي مؤسسة تعليمية أو بحثية أخرى.

DECLARATION

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted elsewhere for any other degree or qualification

Student's name:	اسم الطالب: زكريا خالد زكريا ساق الله
Signature:	التوقيع: [illegible]
Date:	التاريخ: 2015/01/04م

Islamic University of Gaza
Deanery of Higher Studies
Faculty of Information Technology

A Model to Ensure the Integrity of User's Data in the Cloud

Prepared By:

Zakaria Kh. Saqallah
120100859

Supervised by

Dr. Tawfiq S. Barhoom

A Thesis Submitted as Partial Fulfillment of the Requirements for the Degree of Master in Information Technology

2014/1436H

بسم الله الرحمن الرحيم

الجامعة الإسلامية – غزة
The Islamic University - Gaza

مكتب نائب الرئيس للبحث العلمي والدراسات العليا هاتف داخلي: 1150

الرقم Ref ج س غ/35/

التاريخ Date 2014/12/30م

نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة شئون البحث العلمي والدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحث/ **زكريا خالد زكريا ساق الله** لنيل درجة الماجستير في **كلية *تكنولوجيا المعلومات*** **برنامج تكنولوجيا المعلومات** وموضوعها:

مودل للتحقق من سلامة بيانات المستخدمين في السحابة

A model to ensure the integrity of user's Data in the Cloud

وبعد المناقشة العلنية التي تمت اليوم الثلاثاء 08 ربيع أول 1436هـ، الموافق 2014/12/30م الساعة التاسعة والنصف صباحاً، اجتمعت لجنة الحكم على الأطروحة والمكونة من:

د. توفيق سليمان برهوم	**مشرفاً ورئيساً**	[illegible]
د. ربحـي سليمان بركة	**مناقشاً داخلياً**	[illegible]
د. سنـاء وفـا الصايغ	**مناقشاً خارجيًا**	[illegible]

وبعد المداولة أوصت اللجنة بمنح الباحث درجة الماجستير في **كلية *تكنولوجيا المعلومات*/ برنامج تكنولوجيا المعلومات.**

واللجنة إذ تمنحه هذه الدرجة فإنها توصيه بتقوى الله ولزوم طاعته وأن يسخر علمه في خدمة دينه ووطنه.

والله ولي التوفيق ،،،

مساعد نائب الرئيس للبحث العلمي والدراسات العليا

[illegible]

أ.د. فؤاد علي العـاجز

شئون البحث العلمي والدراسات العليا
الجامعة الإسلامية غزة
The Islamic University Gaza
Research & Graduate Affairs

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Abstract

Cloud computing is one of the most popular notions in IT today. It refers to the delivery of computing resources over the Internet. With the development of cloud computing, there are significant concerns and security challenges appear about cloud computing, which will compromise the vision of cloud computing. There are many risks involved with releasing control over user's data because the Cloud service provider could intentionally or accidentally alter or delete some information of the user.

It is not easy for the user to verify that the data is secure, the user need to have confidence in the integrity of his data. In this research, we try to ensure the integrity of user's data in the cloud. We built a model with dynamic data operations support including data append, update and delete.

Online monitoring and offline verification is used also to ensure the integrity of user's data based on checksum verification and local database repository, taking in consideration the size of data file and network bandwidth.

Several experiments were conducted, in order to evaluate the performance depending on execution time. Different size of files were used starting from 11 KB up to 1 GB. We found that each file verification process for one large file (> 5 MB), needs about 0.5 seconds. On the other hand, different types of tests were conducted to evaluate the efficiency of the model in terms of security; a security evaluation proves that our model has been performed at an acceptable level.

Performance and efficiency evaluations show that the model is highly efficient against malicious data modification attack.

Keywords: *Cloud computing, Cloud service provider, Dynamic Data Operations support, Hash function, Online Auditing.*

عنوان البحث باللغة العربية

نموذج للتحقق من سلامة بيانات المستخدم في السحابة

الملخص باللغة العربية

تعتبر الحوسبة السحابية من أكثر مجالات تكنولوجيا المعلومات انتشاراً هذه الأيام، حيث أنها تشير إلى الاستفادة من موارد وإمكانات الحوسبة عبر الإنترنت، ومع تطور الحوسبة السحابية المستمر فإن هناك العديد من المخاوف والتحديات الأمنية الكبيرة حولها التي سوف تؤثر سلباً على زيادة الاستفادة من الحوسبة السحابية مستقبلاً.

هناك العديد من المخاطر التي تنطوي على فقدان المستخدم للسيطرة على بياناته الخاصة لصالح مزود الخدمة السحابية، لأن مزود الخدمة السحابية يمكنه عن قصد أو عن غير قصد تغيير أو حذف بعض المعلومات الخاصة بالمستخدم، لذا ليس من السهل بالنسبة للمستخدم التحقق من أن بياناته الموجودة في السحابة مخزنة في وضع آمن.

في هذا البحث، سنركز على عملية تخزين بيانات المستخدم في السحابة بشكل آمن وذلك لضمان سلامة بيانات المستخدم.

قمنا ببناء نموذج مع توفير دعم للعمليات الديناميكية على البيانات والتي تشتمل على إضافة بيانات جديدة أو تعديل وحذف البيانات الموجودة، وسيدعم أيضاً التحقق من سلامة البيانات أثناء الاتصال بالإنترنت بشكل مباشر، كما ستكون هناك إمكانية للتحقق من سلامة البيانات بعد الانقطاع عن البيانات أي بعد فقدان الاتصال بشبكة الانترنت، وسيتم ذلك بالاعتماد على رموز للتحقق من سلامة الملفات واستخدام قاعدة بيانات محلية أيضاً، مع الأخذ في الاعتبار حجم ملف البيانات وسرعة الاتصال بشبكة الانترنت.

ولتحليل الأداء قمنا بعدة تجارب لقياس وقت التنفيذ، حيث تم استخدام ملفات بأحجام مختلفة تبدأ من 11 كيلو بايت وحتى 1 جيجا بايت، حيث وجدنا أن عملية التحقق من سلامة الملف الواحد الذي يزيد حجمه عن 5 ميجا بايت تستغرق حوالي 0.5 ثانية.

من جهة أخرى قمنا بإجراء عدة اختبارات لتحليل الكفاءة من خلال التحقق من سلامة الملفات، حيث أظهر التحليل أن النتائج بالمستوى المطلوب.

أظهر تحليل الأداء وتحليل الكفاءة أن النموذج المقترح يعمل بكفاءة عالية ضد أي هجوم ضار لتعديل البيانات.

إهداء

إلى من كان له الفضل بعد الله تعالى في وجودي، إلى من علمني وأدبني وصقل شخصيتي،

إلى روحه الطاهرة...إلى أبي

إلى من نذرت عمرها من أجلنا أطال الله في عمرها ... إلى أمي

إلى رفيقة دربي زوجتي أهديها كل طموحي

إلى أخوتي وأبنائي وأساتذتي وأصدقائي وكل من ساندني

Acknowledgment

First and foremost, I thank **Allah Almighty** for helping me to complete this research successfully, and I am asking Allah to guide me to what he loves and pleases.

I would like to express my deep and sincere gratitude to my supervisor, **Dr. Tawfiq S.M. Barhoom**, Dean Faculty of Information Technology, in Islamic University-Gaza, for his guidance and for giving me the opportunity to accomplish this research. He advised me during my work to present the research as clearly as possible.

I would like to extend my thanks to many people, for their advices, especially my **academic staff** in Islamic university, for teaching me, and helping me with my courses in my master's degree.

Special thanks for Dr. Jehad El-Batish, Vice President of Al-Quds Open University, for his special support.

Also, I would like to thank my family: **mother**, **brothers**, **sisters** and a special thank for **my wife**, for her patience during my study.

I cannot forget my colleagues at work, my friends, and all who supported me.

Zakaria Kh. Saqallah

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CSP	Cloud Service Provider.
IaaS	Infrastructure as a Service.
PaaS	Platform as a Service.
SaaS	Software as a Service.
NIST	National Institute of Standards and Technology.
TPV	Third Party Verifier.
TPA	Third Party Auditing.
POR	Proof of Retrievability.
PDP	Provable Data Possession.
MD5	Message Digest Algorithm.
SHA	Secure Hash Algorithm.
ms	Milliseconds

Chapter 1
Introduction

Cloud computing refers to the delivery of computing resources over the Internet. Instead of keeping data on your own hard drive or updating applications for your needs, you can use a service over the Internet to store your information at another location, or you can use its applications.

Cloud computing includes activities such as the use of social networking sites and other forms of computing; however, most of the time cloud computing is concerned by accessing online software applications, data storage and processing power [16]. The importance of Cloud Computing is increasing, it is receiving a growing attention in the scientific and industrial communities, also it is considered a hot researching area of computer network technology.

Cloud Computing is gaining much more popularity because the Cloud providers feel that it is very easy to manage the data in the cloud environment rather than normal web-sites in forms of simple web pages [6]. Cloud computing mainly provides three kinds of services: ***IaaS*** (Infrastructure as a Service), ***PaaS*** (Platform as a Service) and ***SaaS*** (Software as a Service) [6].

The major difference between service based on cloud computing and traditional service is that user's data is stored not in the local server, but in the distributed storage system of the service supplier.

Cloud storage is based on highly virtualized infrastructure and has the same characteristics as cloud computing [35]. In cloud data storage, user stores his data into the cloud server with the help of a Cloud Service Provider [15]. Users can communicate with cloud service providers to retrieve their data. In addition, users can perform operations on the cloud such as insert, delete and update.

The movement of data to centralized services could affect the security of user's interactions with the files stored in cloud storage. There are many risks involved with releasing control over user's data. Data may be damaged while in transit to or from the storage provider. Additionally, the Cloud service provider could intentionally or accidentally alter or delete some information from the cloud server. Because the user does not know, how the cloud provider control or monitor his data. Hence, the system must have some sort of mechanism to ensure the data integrity.

The current Cloud security based on the assumption that the user should trust the provider, but it is not enough, so in order to ensure the integrity of data in Cloud, efficient methods that enable on-demand data integrity verification have to be designed. Hence, the verification of data integrity must be conduct without explicit knowledge of the entire data files [14].

In this research, we introduce a model to ensure the integrity of user's data in the cloud. We introduce a model with dynamic data operations support including data update, append and delete. Online monitoring and offline verification is used also to ensure the integrity of user's data in the cloud based on checksum verification and local database repository, with respect to the size of data file and network bandwidth.

We built a tool called Integrity-Verifier which is written in Visual Basic.NET language, this tool consists of six classes, file preparation class, checksum verifier, store class, online monitor, offline verification and operations class. These classes are combined together to ensure the integrity of user's data in the cloud.

We conducted several experiments, to evaluate the performance depending on execution time. To accomplish our experiment, we collected a dataset, our dataset contained different file sizes and types, so we collected large amount of data files with 9.16 GB. The size of these files starting from 11 KB up to 1 GB. We found that each file verification process for one large file (> 5 MB), needs about 0.5 seconds.

We found also that offline verification is the fastest method in our model, the verification time is less than 13 second for all data files in our dataset which contains 1917 file. On the other hand, several types of tests were conducted to evaluate the efficiency of the model in terms of security; a security evaluation proves that our model has been performed at an acceptable level.

1.1 Statement of the problem

The correctness of the user's data files being stored on the cloud servers must be guaranteed. It is very important to detect any inconsistencies and ensuring the integrity of user's data, which is stored in the cloud without keeping the original data locally in user side. From the perspective of data security, it is an integrity problem.

1.2 Objectives

In this section, we present main objective and specific objectives of the research work.

1.2.1 Main objective

The main objective of this research is to build a model with dynamic data operations support (data append, update and delete), to ensure the integrity of user's data in the cloud based on checksum verification, offline verification and online auditing.

1.2.2 Specific objectives

i. Using appropriate algorithm to generate meta-data for checksum verification before storing a file to the cloud.
ii. Upload the user's data files to the cloud and saving some data information to a local database.
iii. Design a model which contains the support for dynamic data operations and online monitoring.
iv. Implement the model and conduct the experiment in a cloud environment.
v. Evaluate the performance and the efficiency of the model in term of execution time and security efficiency.

1.3 Importance of the thesis

The leading U.S. market research firm Gartner released a report “Assessing the Security Risks of Cloud Computing”, this report stated that cloud computing has great risks on data integrity, data recovery and privacy, etc. [39].

Cloud computing is an unsecure platform from the perspective of the cloud users and cloud storage is a rich resource for both hackers and national security agencies.

Therefore, the main importance of this research rises from:

i. This research should alleviate much of today's users fear of cloud computing.

ii. Integrity check can be performed without the need to save a copy of the data locally.

1.4 Scope and Limitations of the thesis

We aim to build a model to ensure the integrity of user's data in the cloud, the research is focused specifically on integrity security problem, the model will allow one user to verify his data, this model will not allow third party verifiers to check the integrity of the stored data, also the usage of the cloud has some limitations and these limitations could be the same limitations we expect to face in this research in the cloud environment such as:

1. ***Access to storage media*****:** the data, which is captured and stored over years in the company archives. The ability to verify old data.
2. **Storage Limit:** the service provider gives a free just limit storage size also limited file size. Some companies make limitation for the one file size to make control.
3. **Security issues:** some providers put advertisements in user's side to make some money so the user's data file may not be secured before preparing and storing it in the cloud.
4. ***The flexibility & extensibility*** of the cloud make the ensuring of user's data very hard, when cloud service provider want to move the user's data files from one server to a new one, it may appears integrity problem.

1.5 Methodology

To accomplish the objectives of this research, the following methodology will be followed:

1- Reviewing the recent researchs of ensuring the integrity of user's data in the cloud that is related to our problem. Studying the existing approaches, and noting the advantages and disadvantages of each method in order to overcome in our research. When we reviewed most previous studies we found that there are very important contributions that we can introduce in this research.

2- Process the proposed model which contains compute verification tokens, Dynamic Data Operation Support, integrity verification using checksum verification, local database repository and online auditing.

   Unlike previous studies which we mentioned in Chapter 3, we are going to build the model, and we will consider these issues:

   a. Verifying data integrity without the need to keep a copy of data locally.
   b. Working with large files, without the need to read whole file.
   c. Detect missing files, when any file is deleted by unauthorized.
   d. Detecting unauthorized modifications online.
   e. Support dynamic data operations (modify, delete, and add), that user can access and modify his data.

   Also we search about appropriate algorithms that can be used to prepare the file before storing in the cloud, we will use a universal hash function called SHA256, and we will explain and justify the choosing of this algorithm in details.

3- Implement the model, File Preparation will implemented. This will conduct using VB.NET.

4- Evaluate the performance and the efficiency of the model in term of execution time and security efficiency.

5- Analyzing the obtained results and justify the model feasibility.

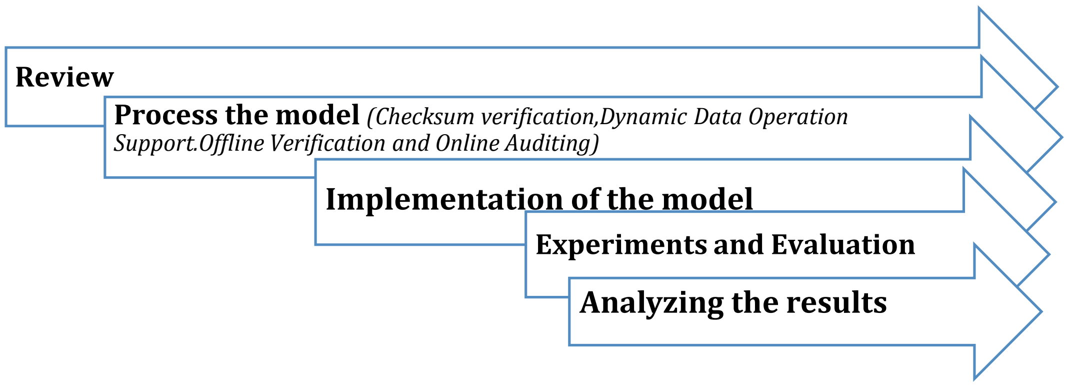

Review
Process the model (Checksum verification,Dynamic Data Operation Support.Offline Verification and Online Auditing)
Implementation of the model
Experiments and Evaluation
Analyzing the results

Figure 1.1: Research Methodology

1.6 Thesis structure

This thesis consists of six chapters: Introduction, Theory Background, Related Works, Data integrity model, Experiments and Evaluation and finally, Conclusions and Future work.

The main points discussed in the chapters are listed below:

i. **Chapter 1:** ***Introduction***.

ii. **Chapter 2:** ***Theoretical Background***, presents an overview about cloud computing.

iii. **Chapter 3:** ***Related Works***

iv. **Chapter 4:** ***Data Integrity Model,*** model architecture and implementation.

v. **Chapter 5:** ***Experiments and Evaluation:*** the experimental works, starting from preparing user's data files and ending with verifying files on cloud side, evaluating and analyzing the experimental results.

vi. **Chapter 6:** ***Conclusions and Future Work:*** presents conclusions and possible future works.

Chapter 2
Theoretical Background

In this chapter we present an overview of cloud computing, its definition, essential characteristics, deployments model and cloud architecture key components. We also present cloud computing benefits, challenges and security issues.

2.1 Cloud computing overview

Cloud computing is the delivery of computing services over the Internet. Cloud services allow individuals and businesses to use software and hardware that are managed by third parties at remote locations. Examples of cloud services include online file storage, social networking sites, webmail, and online business applications.

The following definition of cloud computing has been developed by the U.S. National Institute of Standards and Technology (NIST):

"*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models*" [1].

2.1.1 Essential characteristics

The cloud model is composed of five essential characteristics identified by NIST as follows [1]:

Figure 2.1: Five Essential Characteristics of Cloud Computing [10]

1. ***On-demand self-service***. A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider

2. ***Broad network access.*** Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).

Figure **2.2** *Thin and Fat*

3. ***Resource pooling***. The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.

4. ***Rapid elasticity.*** Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provision often appear to be unlimited and can be appropriated in any quantity at any time.

5. ***Measured service***. Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

2.1.2 Deployment models

Deploying cloud computing can differ depending on requirements, each with specific characteristics that support the needs of services and users of the clouds in particular ways. Following are the four types of cloud deployment models identified by NIST [1].

1- ***Private Cloud.*** The cloud infrastructure has been deployed, and is maintained and operated for a specific organization. The operation may be in-house or with a third party on the premises.

2- ***Community Cloud.*** The cloud infrastructure is shared among a number of organizations with similar interests and requirements. This may help limit the capital expenditure costs for its establishment as the costs are shared among the organizations. The operation may be in-house or with a third party on the premises.

3- ***Public Cloud.*** The cloud infrastructure is available to the public on a commercial basis by a cloud service provider. This enables a consumer to develop and deploy a service in the cloud with very little financial outlay compared to the capital expenditure requirements normally associated with other deployment options.

4- ***Hybrid Cloud.*** The cloud infrastructure consists of a number of clouds of any type, but the clouds have the ability through their interfaces to allow data and/or applications to be moved from one cloud to another. This can be a combination of private and public clouds that support the requirement to retain some data in an organization, and also the need to offer services in the cloud.

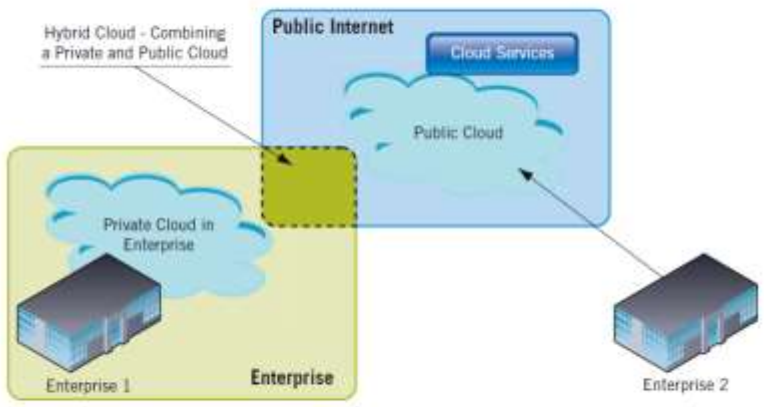

Hybrid Cloud - Combining a Private and Public Cloud
Public Internet
Cloud Services
Public Cloud
Private Cloud in Enterprise
Enterprise 1
Enterprise
Enterprise 2

Figure 2.3: Private, Public and Hybrid Cloud Deployment [8]

2.1.3 Service models

Cloud computing mainly provides three kinds of services: ***IaaS*** (Infrastructure as a Service), ***PaaS*** (Platform as a Service) and ***SaaS*** (Software as a Service). The major difference between service based on cloud computing and traditional service is that user data is stored not in the local server, but in the distributed storage system of the service supplier.

1- ***Infrastructure as a Service (IaaS).*** The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

2- ***Platform as a Service (PaaS).*** The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

3- ***Software as a Service (SaaS).*** The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

2.1.4 Cloud architecture key components

In the cloud architecture, key components are Cloud Service Provider, User (Data Owner) and Third Party Verifier.

1- ***Cloud Service Provider (CSP):*** CSP has significant resources.

2- ***User:*** User may be a person or an organization who have data to be stored in the cloud and rely on the cloud for data computation.

3- ***Third Party Verifier (TPV):*** TPV has expertise and capabilities that users may not have.

2.2 Cloud computing benefits

Many of the benefits when using Cloud Computing are the lower costs associated [12]. The following are some of the possible benefits for those who offer cloud computing-based services and applications [24]:

2.2.1 *Cost savings.*

Companies can reduce their capital expenditures and use operational expenditures for increasing their computing capabilities. The provider is responsible for software deployment and maintenance with his own infrastructure. The user only pays for technical support [12].

2.2.2 Reliability.

Services using multiple redundant sites can support business continuity and disaster recovery.

2.2.3 Scalability/Flexibility.

Companies can start with a small deployment and grow to a large deployment fairly rapidly, and then scale back if necessary. Also, the flexibility of cloud computing allows companies to use extra resources at peak times, enabling them to satisfy consumer demands. Resources within the cloud can be treated as an `unlimited' medium [12].

2.2.4 Maintenance.

Cloud service providers do the system maintenance, and access is through APIs that do not require application installations onto PCs, thus reducing maintenance is required.

2.2.5 Mobile accessible.

Mobile workers have increased productivity due to systems accessible in an infrastructure available from anywhere.

2.3 Cloud challenges and security issues

In Cloud computing there are some challenges and security issues, we will introduce some of notable challenges and security issues:

2.3.1 Cloud computing challenges

The following are some of the notable challenges associated with cloud computing [24]:

- **Security and *Privacy***. Perhaps two of the more “hot button” issues surrounding cloud computing relate to storing and securing data, and monitoring the use of the cloud by the service providers. These issues are generally attributed to slowing the deployment of cloud services.
- **Lack *of standards***. Clouds have documented interfaces; however, no standards are associated with these, and thus it is unlikely that most clouds will be interoperable.

- **Continuously *evolving.*** User requirements are continuously evolving, as are the requirements for interfaces, networking, and storage. This means that a “cloud,” is also continuously evolving.
- **Compliance *concerns.*** Legal and Regulatory Issues**,** in some of European countries, Government regulations do not allow users to store their personal data outside the state or country [20].

  In order to solve this issue, cloud providers need to setup a data center within the country to validate these regulations. But it is difficult to do that, so it considered as a big challenge for cloud providers. It is necessary for Cloud provider to discuss that with experts in service negotiation, distributed services and risk assessment for that.

2.3.2 Cloud computing security issues

Cloud computing grows very fast. Many users stores their data on Cloud data storage.

A user stores his data through a Cloud Service Provider into a set of cloud servers called cloud data storage, the user interacts with the cloud servers via CSP to access or retrieve his data and the user may need to perform dynamic data operations on his data.

Data security is becoming a fundamental obstruction in cloud computing, the three quality aspects of information security, confidentiality, integrity and availability, the aspect of confidentiality gets most of the attention. Indeed potential breaches of information security attract a lot of attention. About integrity of information, however, little has been published [23].

2.3.2.1 Data integrity problem

One important security problem is guaranteeing the integrity of remotely stored data. Data integrity is defined as the accuracy and consistency of stored data, in absence of any alteration to the data between two updates of a file or record [22]. Cloud services should ensure data integrity and provide trust to the user privacy.

Data integrity in the Cloud system means to preserve information integrity [29] (i.e., not lost or modified by unauthorized users). As data is the base for providing Cloud Computing services, such as Data as a Service, Software as a Service, Platform as a Service, keeping data integrity is a fundamental task [35]. Integrity monitoring is essential in cloud storage for the same reasons that data integrity is critical for any data center. Data corruption can happen at any level of storage and with any type of media.

2.3.2.2 Cloud storage

Cloud storage is a model with virtualized pools of storage, which is generally operated with large third party data centers, distributed and spread across different geographical locations [3]. Controller failures and loss of bits of data on storage media are examples of corruption. The truth is that data corruption can happen in a storage environment [4]. Cloud storage are still data centers, with hardware and software, and are still vulnerable to data corruption.

It is of critical importance to assure users data are consistent and correctly stored without the existence of local copies.

a. **Storage correctness** is to ensure user's data are stored appropriately and kept intact all the time in the distributed servers in the cloud [5].
b. **Data consistency** requires that all relationships between data items and replicas are as they should be, i.e., that the data representation is correct.

Summary

We presented in this chapter a cloud computing overview, its definition, essential characteristics, deployments model and Cloud architecture key components. In addition, we presented cloud computing benefits, challenges and security issues.

We discussed the security problems in the cloud and declared that data security is becoming a fundamental obstruction in Cloud Computing. Data integrity is critical for any storage center, and it is important to users to ensure that their data is stored securely without the existence of local copies.

Chapter 3
Related Works

When we discuss cloud computing issues, many threats are raised. One of the major security issues is data integrity. A lot of studies discussed this problem and introduced some solutions to ensure data integrity in the cloud. In this chapter, different related works are reviewed. Some of them can be a basis for helping in our data integrity model.

Some works study the problem of cloud storage correctness to ensure data integrity, others focus on collecting some bytes from each file in order to make the verification process very fast with respect to network bandwidth, and others argue that allowing third party auditors is important.

So, we classify related works according to following categories:

1- Storage correctness verification.
2- External auditing.
3- File verification.

For each category, we introduce some related works, noting the advantages and disadvantages of each one. Finally we introduce a conclusion in order to overcome the disadvantages in our research.

1- Storage correctness verification

Ari Juels et al. [2] introduce "proof of retrievability" (POR) model for ensuring the remote data integrity, using error correcting code spot-checking, to ensure both "possession" and "retrievability" of data files on archive service systems. Specifically, a blocks called "sentinels" are embedded randomly into a data file, then the file is encrypted to protect the positions of these special blocks. This scheme involves the encryption of the file F using a secret key.

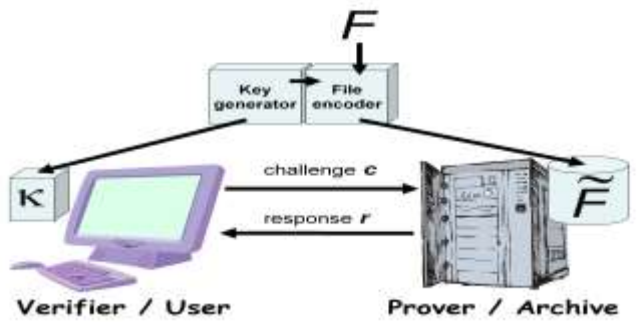


Figure 3.1: Schematic of a POR System. An Encoding Algorithm Transforms a Raw File F Into an Encoded File F˜ [2]

It becomes computationally cumbersome especially when the data to be encrypted is large. Also it is not supporting dynamic data operations because it works on static data.

An improved framework for POR proposed by **Kevin D. *Bowers et al.*** [14] for distributed systems. The effectiveness of this scheme rests primarily on the preprocessing steps that user conducts before outsourcing the data file. Any change of file content, must propagate through the error-correcting code, introducing computation and communication complexity and their scheme is focusing on static data.

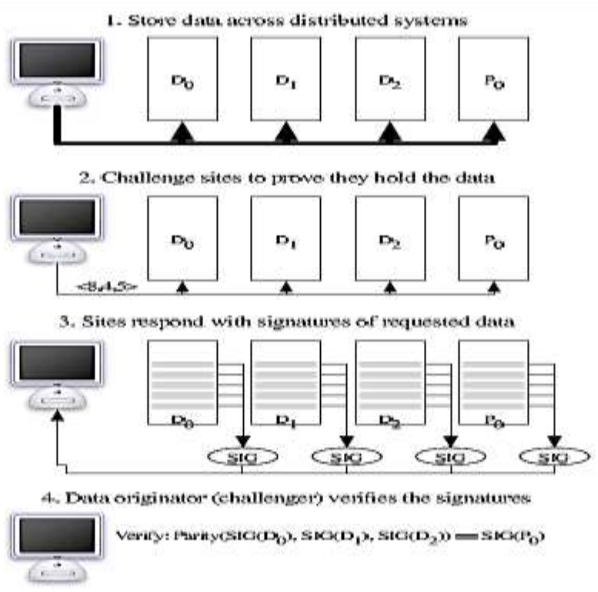

1. Store data across distributed systems
D0
D1
D2
P0
2. Challenge sites to prove they hold the data
<3,4,5>
3. Sites respond with signatures of requested data
SIG
4. Data originator (challenger) verifies the signatures
Verify: Parity(SIG(D0), SIG(D1), SIG(D2)) == SIG(P0)

Figure 3.2: Basic Storage, Signature Request and Verification Protocol [14]

On the other hand, **Thomas Schwarz et. al.** [30] proposed to ensure file integrity across multiple distributed servers, using m/n erasure-correcting coding to safeguard the stored data and use algebraic signatures hash functions with algebraic properties for verification.

Their scheme has some advantages. First, it uses only small messages for verification; second, it allows verification without the need to compare against the original data.

Their scheme uses only small messages for verification. So it is very fast, the use of algebraic signatures will permit the construction of large-scale distributed storage systems in which large amounts of storage can be verified with minimal network bandwidth.
However, their schemes only considers static data files, algebraic signatures are not secure like cryptographically secure hash functions of secure checksums.

Giuseppe Ateniese et. al. *[11],* constructed Provable Data Possession (PDP) technique based on symmetric key cryptography, their technique allows outsourcing of dynamic data, their contribution is in two-folds, first on the efficiency, the proposed PDP scheme relies only on efficient symmetric key operations in both setups (performed once) and verification phases. Second it supports dynamic data operations, including: modification, deletion and appending. This technique, while can be useful to ensure the storage correctness without having users possessing data, cannot address all security threats in cloud data storage, since they works on whole data, and did not work file by file, also dynamic data operations they presented is not efficient because we need every time to read verification tokens from server and upload it after any dynamic operation.

2- External auditing

Some researchers proposed allowing a third-party auditing TPA to help the customers and to keep online storage honest, ***Mehul A. Shah et. al***. [17], described approaches that support auditing of online storage, they argue that allowing TPA is important for online service-oriented economy, it increases the efficiency of mitigation risk. This is done by encrypting the data then sending a number of pre-computed symmetric-keyed hashes over the encrypted data to the auditor. They present issues about internal and external auditing and detail ways of auditing online storage services. The auditor repeatedly checks the stored contents.

However, their scheme only works for encrypted files, and auditing process takes long time, and not studying the problem of dynamic data operations.

Zhang lianhong, Chen Hua [41]. Present a scheme, which enables a third-party verifier to check data integrity based on RSA assumption. Their scheme cannot realize the dynamics data for remote data integrity check, and it works with fixed state, in the other hand it does not mentioned about different file size in verification phase.

3- File verification

Sravan Kumar R, Saxena, A. [28] present a scheme which does not involve the encryption of the whole data. But only few bits of data per data block to reduce computational overhead on clients side. In data integrity protocol the verifier needs to store a single cryptographic key whatever the size of the data file F. The verifier does not store any data with it but appends some Meta data to the file and stores it at the archive, the verifier uses this Meta data to verify the integrity of the data.

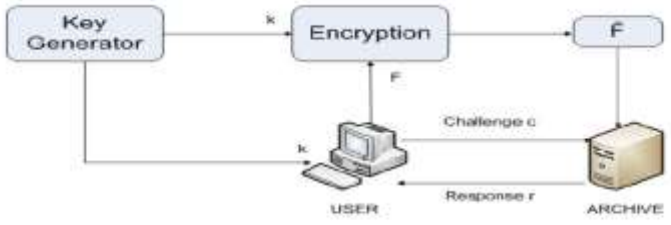


Figure 3.3: Schematic View of a Proof of Retrievability Based on Inserting Random Sentinels in the Data File F [28]

Their scheme applies only to static storage of data. It cannot handle the case when the data need to be dynamically changed. In addition, the number of bits that they depend on is not enough with respect to the files size; it seems to be that the probability to detect any modification in large files is not high.

Another study, ***Saranya Eswaran and Dr.Sunitha Abburu***, [26], aimed to ensuring the integrity of the data and provided the proof that data is secured, using cryptographic key to secure the data in the cloud. Their approach depending on TPA, they argue that user does not has experience like TPA to check the integrity of his data, and it is difficult for user to check. However, their approach does not discuss the file size and dynamic data operations.

Another study using hash algorithm, ***Selman Haxhijaha et.al***. [27], they used MD5 hash algorithm. First, the user pre-computes the hash value of the file, sends the file to the cloud and stored the computed hash value in local repository.

User can verify his data integrity by, retrieve the file from cloud, computes hash value of the file again and then matches it with the stored hash values at local hash repository.

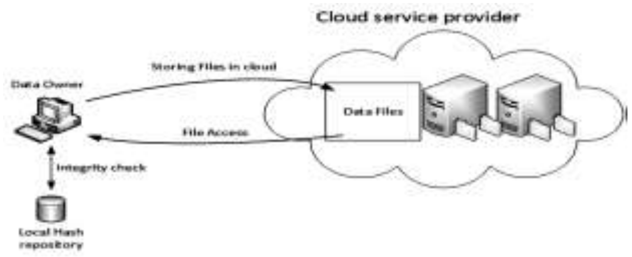


Figure 3.4: Data Integrity Check Using Hash Functions [27]

However, it is not efficient when we want to verify the integrity of the file to download or read whole file to local side, especially with large files, also they use MD5 algorithm which cannot resist collision attack.

Summary

Most prior works mentioned in this chapter, introduced some solutions, but it is not enough, in [14], [2], [30], they works on static data files (like backups) and not supporting dynamic data operation. In [17], [27] introducing significant computation and communication complexity. The rest of the related works does not consider the size of the file.

In this research, we considered all these issues, and we introduce also online monitoring.

Chapter 4
Data Integrity Model

In this chapter we presented our model in Section 4.1, to achieve the specific objectives of our research, then we described in details the implementation in Section 4.2.

4.1 The proposed data integrity model

Our proposed model architecture is shown in Figure 4.1, which contains: preparing the file, Dynamic Data Operation Support, checksum verification on each file, online auditing and offline verification.

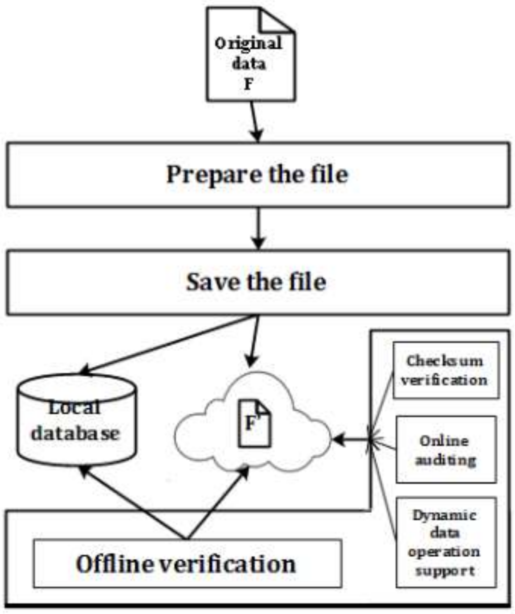


Figure 4.1: The Proposed Model Architecture

We aim to design efficient mechanisms to achieve the verification goal on the file directly without the need of keeping the old file locally, the client should create suitable meta data of the file F before sending and storing it to the cloud, this meta data will be used in the later stage of verification the data integrity at the cloud storage.

The proposed model flowchart shown in figure 4.2:

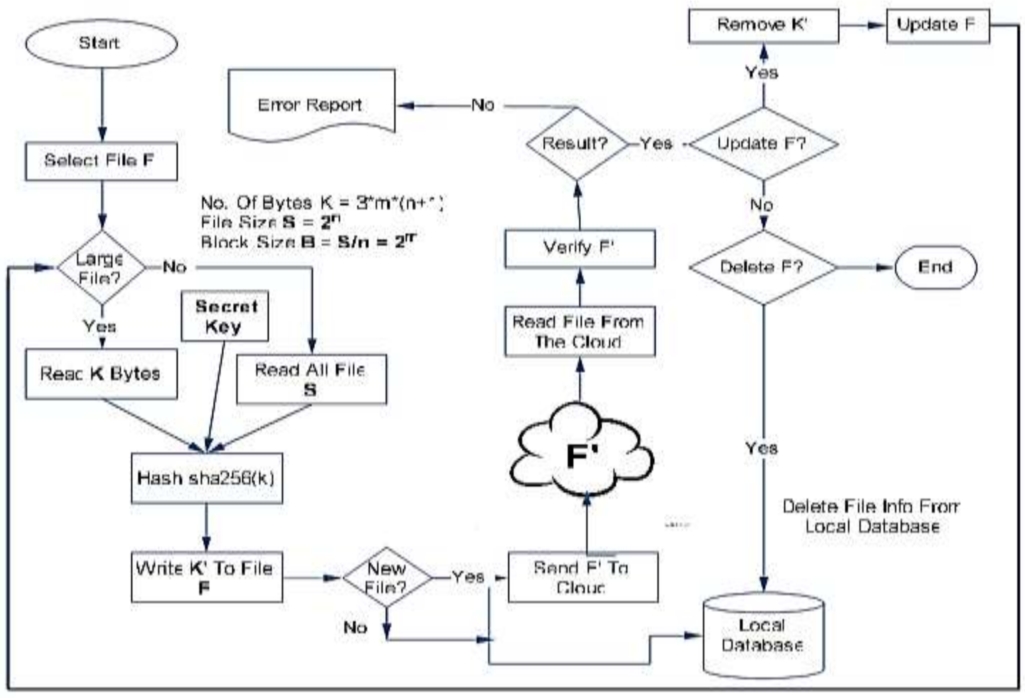


Figure 4.2: The Proposed Model Flowchart

When checking for data integrity the client queries the cloud storage for suitable replies based on which it concludes the integrity of its data stored in the client. In order to achieve that, the model will consist of the following methods:

4.1.1 File preparation step

Prepare the file before storing it into the cloud, this is done by generating and encrypt the meta-data then writing it to the file, in Figure 4.2 we can see the following steps which illustrates the file preparation method:

Step 1: Generation of meta-data

Before generating meta-data to be appended it at the end of the data file, we will take some consideration especially for large data files such as:

- Collect special bytes from important positions in a data file, such as the start and end bytes of any large file, these considerations include:
- Collect a reasonable number of bytes with respect to the size of data file (in bytes), it is better to collect large number of bytes as possible, this is necessary to increase the probability of detection of any unauthorized

modifications, it is not possible to choose the same number of bytes of different file size.

- Network bandwidth must be minimized, as the size of the proof is comparatively very small compared to the file size.
- For small files (less than 0.5 MB -this size is determined according to our experiment resources (see Chapter 5)), it is better to collect all bytes of the file.
- Finally, choose an appropriate hash algorithm to encrypt the collected bytes, then append the resulting hash to the end of the file with reasonable number of bytes.

As we mentioned above this method takes in consideration the size of the file, which means the number of bytes will change according to the size of the file.

Step 2: Collecting bytes

To do this, let the user wishes to store file F, which consists of n + m file blocks, where m is a value to detect any change in the file size. We preprocess the file and create meta-data to be appended to the end of a file. Let each of the n + m data blocks have BS bytes in them. K will be the number of bytes we will get from each block. The following steps show how to calculate and extract the bytes number from large files:

1- *Extract and determine number of blocks*

Extract the value of n from file size (F) in bytes, $FS \approx 2^n$, we extract the result of this equation:

- *n= integer value of* $(Log_2(FS)/log_2(2))$

2- *Detect any change on the file size*

This is important to detect any change of file size, we will use this value to determine the number of blocks of the file.

- $m = Fs$ ***mod*** 3

3- *Calculate the number of blocks of the file*

- *Number of blocks =* $n + m$

4- *Calculate block size*

- *Block size (BS)= integer value of (FS / (n + m))*

5- *Calculate the number of required meta-data with size A*

Extract the value of L from block size, BS ≈ 2^L, we extract the result of this equation:

- *L= integer value of (Log$_2$ (BS)/log$_2$ (2))*
- *K=3 * L. (Collect L bytes from 3 places in each block, start, middle and the end of the block). we can see this in Figure 4.3*
- *A = K * (n + m).*

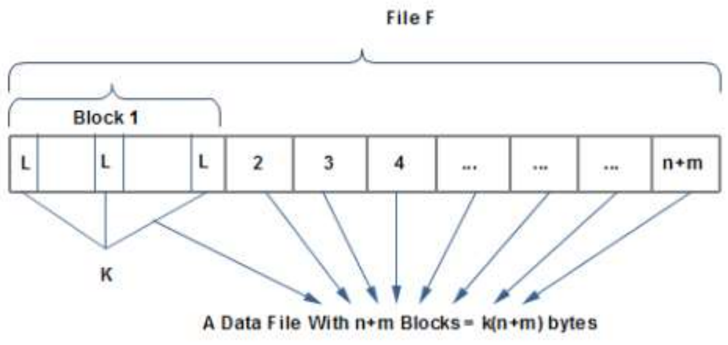


Figure 4.3: Collecting Bytes From File ***F***

For small files (less than 0.5 MB), we will collect all bytes in data file, it is better to collect largest number of bytes from the file, with my test results, the performance with this size is suitable with respect to our resources and network bandwidth.

In Table 4.1, we can see examples of different meta-data size according to different files size.

Table 4.1: Size of Meta-Data Depending on the File Size

File Size(B)	**n**	**m**	**Block Size**	**L**	**K**	**A Meta-Data**
5,242880	22	2	218453.333	17	51	1224
31,457280	25	0	1258291.2	20	60	1500
157,286400	27	0	5825422.22	22	66	1782
262,1440000	31	1	81920000	26	78	2496

Step 3: Inserting a secret key

We can also add a secret key or password (some bytes) to these bytes in order to harden against any malicious attack, this is known by the user only. As shown in Figure 4.2 we added a secret key before the encryption step.

Step 4: Encrypting collected bytes

After collecting the bytes, we need to encrypt these bytes using a hash function. A hash function is any function that can be used to map digital data of arbitrary size to digital data of fixed size, with slight differences in input data producing very big differences in output data [36].

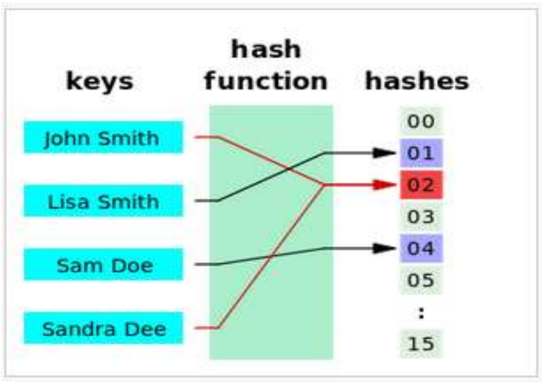


Figure 4.4: A Hash Function that Maps Names to Integers From 0 to 15 [36].

The values returned by a hash function are called hash values, hash codes, hash sums, or simply hashes.

We can understand the idea of hash digital data of arbitrary size to digital data of fixed size simply in Figure 4.4.

A hash value can be used to uniquely identify secret information. This requires that the hash function is ***collision resistant***, which means that it is very hard to find data that generate the same hash value [36].

A hash algorithm example is SHA-1, one of the most widely used cryptographic hash functions, generates 160 bit values. A file hashed with SHA-1 could look like:

752c14ea195c369bac3c3b7896975ee9fd15eeb7.

A collision is an attack specific to hash [31], when 2 different files produce an identical hash. It is then possible to substitute a file for another. In our domain of expertise we could then imagine to replace an official certificate by a fraudulent one having the same hash values. SHA-0 is not resistant to collision attacks that is the reason why it is not used anymore.

As computing power has increased the feasibility of breaking the SHA1 hash algorithm increased. Plans within the industry have been made to transition from SHA1 to SHA256 (SHA2). However with recent announcements from Microsoft and Google about depreciating support for SHA1 in browsers this transition has been accelerated [7].

In 2005, security flaws were identified in SHA-1, namely that a mathematical weakness might exist, indicating that a stronger hash function would be desirable. Although SHA-2 bears some similarity to the SHA-1 algorithm, these attacks have not been successfully extended to SHA-2 [37].

SHA-2: A family of two similar hash functions, with different block sizes, known as SHA-256 and SHA-512. They differ in the word size; SHA-256 uses 32-bit words where SHA-512 uses 64-bit words [38].

So, we will use SHA-256 to generate a hash value of bytes collected in previous step.

Step 5: Writing encrypted meta-data A to the file

After computing the hash value of collected bytes using SHA-256 hash function algorithm, we will write the result to the end of the file F, The size will be 64 byte.

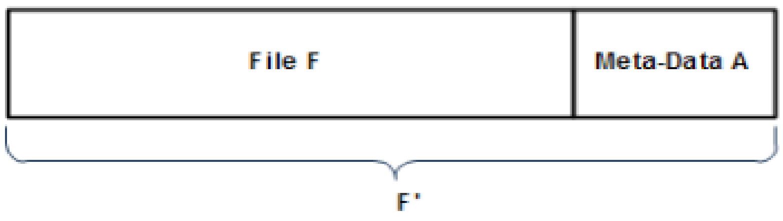


Figure 4.5: Writing Meta-Data A to Original File F

This step is taking more time than computing the hash value especially in large files, because it is done by opening the file to write the meta-data at the end of it then closing the file, but this time is acceptable.

Step 6: Sending the file to the cloud

After appending the meta-data into the file F, it becomes encrypted and ready to store into the cloud server as an encrypted file F'. At this step we store some file information to a local database repository. This is very important to verify the missed files especially when we are going offline and disconnect with the cloud. The information needed will be:

1- File previous path.

2- File new path (in the cloud).

3- File name.

4- File size.

5- Last modification date of the file.

We can use this information when we go online again. The time required to send the file to the cloud depends on the upload speed of the available network bandwidth.

4.1.2 Dynamic operation support

In cloud data storage, there are some scenarios where the data stored in the cloud is dynamic, like photos, documents etc. Therefore, it is important to consider the dynamic data operation support including (add, update and delete) on the data file.

1. **Update operation**

   In cloud data storage, sometimes the user need to modify his data files directly in the cloud, the model allows the user to do this by:

   1- Verify the integrity of the file F.

   2- Remove the old hash value from the end of the file.

   3- User can modify whatever he wants on his data file.

   4- Re-compute verification hash value and append the new meta-data to the end of the file.

   5- Update the local database repository for the new information of the file.

2. **Add operation**

   The model will allow the user to add any new data files to his own storage media files, by repeating the same steps mentioned before in the file preparation Section 4.1.1, and then store the file information to the local database repository. This operation is necessary to ensure that the user can work with his own storage whenever needed.

3. **Delete operation**

   The user may wish to delete his own data files from the cloud servers; he can accomplish this operation simply by deleting the data file and delete its information from the local database repository. This operation is important, so the user can differentiate between his own operations on the work of others.

4.1.3 Checksum verification operation

It is important to ensure the integrity of user's data in the cloud to achieve the purpose of the thesis, this operation is accomplished by:

a. Specify the file F' to be verified.
b. Read the hash value stored at the end of the file (old-hash).

c. Re-compute the generation of the hash value (new-hash) as explained in Section 4.1.1 on the file F' without old-hash value.
d. Match the old-hash value with new-hash, if the old-hash value matches then the file is intact, if not, then the file is tampered and its integrity is compromised.

4.1.4 Offline verification

Another verification is to compare file information with local information which stored in local database repository, any change of file size or last date modification, or if the file is missing, means the integrity of the file is compromised.

4.1.5 Online auditing

Recently most users can enjoy their online connectivity to the internet, so they can follow their social media and access all the time to the internet, we can use this feature to introduce online watcher to user's data (files) which are stored in cloud servers.

Cloud providers provide desktop applications that allow users to access files on file explorer in any operating system, and allow users to synchronize their files to their computers locally.

We use this features to allow the user to audit his data online, by watching any change on important attributes of any file information, such as file size and last modification date on any file, which is stored in any folder or sub folders of user's data. This method allow user to monitor his data directly and online, to detect any unauthorized modifications.

4.2 Implementation

In this section, we present our model implementation. We describe the tool which we built, programing language and main classes.

We build a tool called Integrity-Verifier which is written in VB.NET 2013 language. VB.Net is a simple, modern, object-oriented computer programming language developed by Microsoft to combine the power of .NET framework and the common language runtime with the productivity benefits that are the hallmark of Visual Basic [32].

We use VB.NET because it is easy to learn, structured language, produces efficient programs and it can be compiled on a variety of computer platforms.

4.2.1 Integrity-Verifier main classes

Integrity-Verifier tool consists of six classes as in Figure 4.6, file preparation class, checksum verifier, store class, online monitor, offline verification and dynamic operations class. These classes are combined together to ensure the integrity of user's data in the cloud. In next, classes are described in more details:

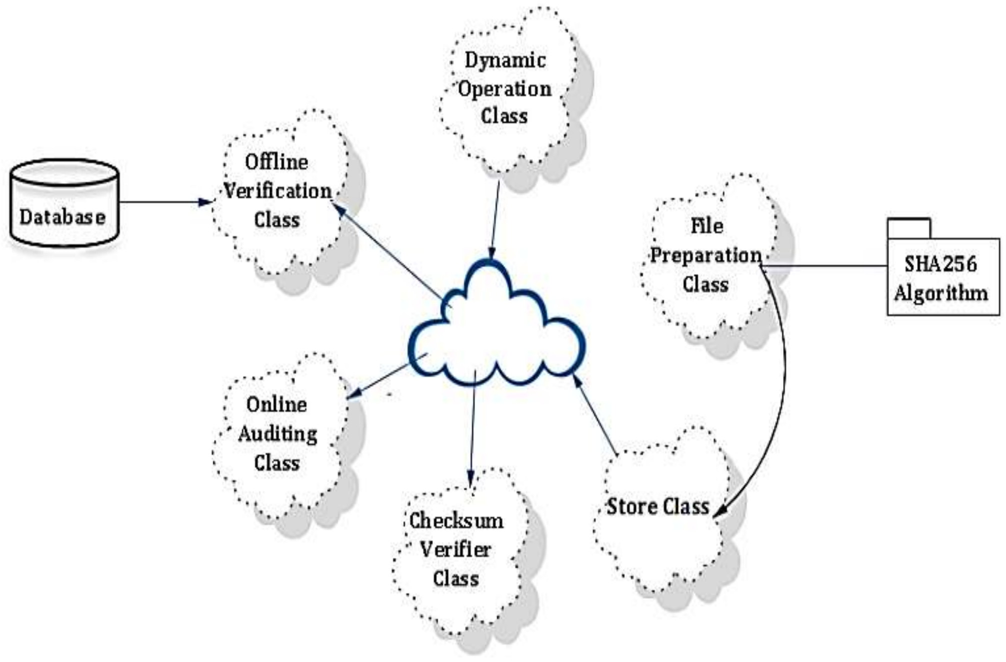
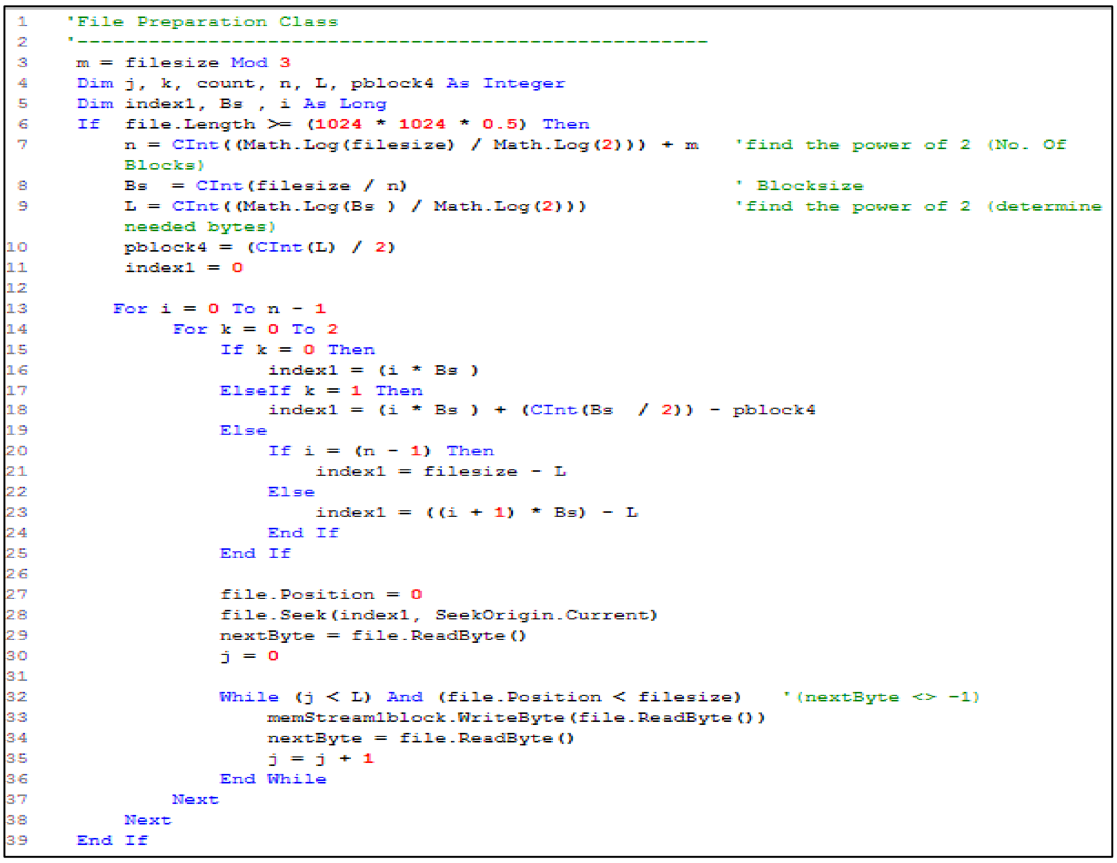


Figure 4.6: Main Classed of Integrity-Verifier

1. File Preparation Class

Preparing the large file before sending it to the cloud storage, in Figure 4.7, we can see the file preparation class, and we can see our calculations, which are mentioned in Section 4.1 in details.



```
'File Preparation Class
'-----------------------------------------------------
 m = filesize Mod 3
 Dim j, k, count, n, L, pblock4 As Integer
 Dim index1, Bs , i As Long
 If  file.Length >= (1024 * 1024 * 0.5) Then
     n = CInt((Math.Log(filesize) / Math.Log(2))) + m   'find the power of 2 (No. Of
     Blocks)
     Bs  = CInt(filesize / n)                           ' Blocksize
     L = CInt((Math.Log(Bs ) / Math.Log(2)))           'find the power of 2 (determine
     needed bytes)
     pblock4 = (CInt(L) / 2)
     index1 = 0

    For i = 0 To n - 1
         For k = 0 To 2
             If k = 0 Then
                 index1 = (i * Bs )
             ElseIf k = 1 Then
                 index1 = (i * Bs ) + (CInt(Bs  / 2)) - pblock4
             Else
                 If i = (n - 1) Then
                     index1 = filesize - L
                 Else
                     index1 = ((i + 1) * Bs) - L
                 End If
             End If

             file.Position = 0
             file.Seek(index1, SeekOrigin.Current)
             nextByte = file.ReadByte()
             j = 0

             While (j < L) And (file.Position < filesize)    '(nextByte <> -1)
                 memStream1block.WriteByte(file.ReadByte())
                 nextByte = file.ReadByte()
                 j = j + 1
             End While
         Next
     Next
 End If
```

Figure 4.7: A Snapshot of the File Preparation Class

We can prepare the file by these steps:

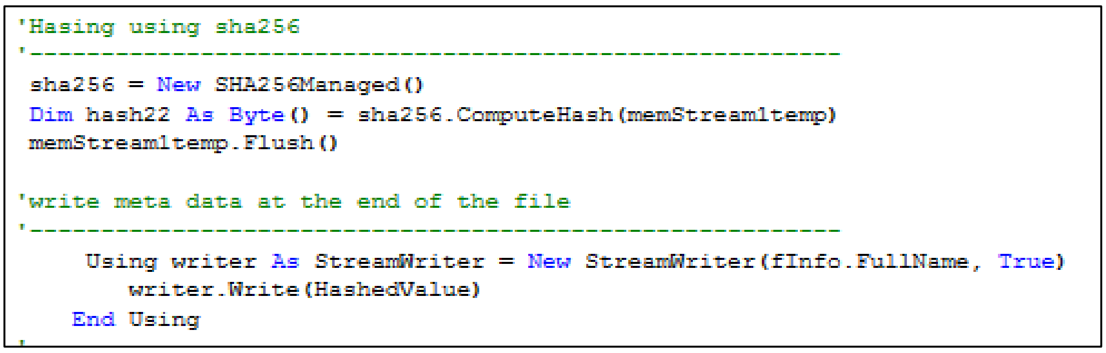
Step 1: Select a file.

Step 2: The system will check whether the file size is large or small, we decided before that if file size is greater than 0.5 MB, it will be considered as a large file,

Step 3: Collecting the necessary bytes as mentioned before in section 4.1, calculate the hash value using SHA-256 algorithm, and write the result value at the end of the file.

Step 4: Append a secret key, known by the user in order to increase the security.

Step 5: Encrypt the collected bytes, using hash function called SHA256, as we can see in Figure 4.8, we can see a snapshot of hashing the collected bytes.

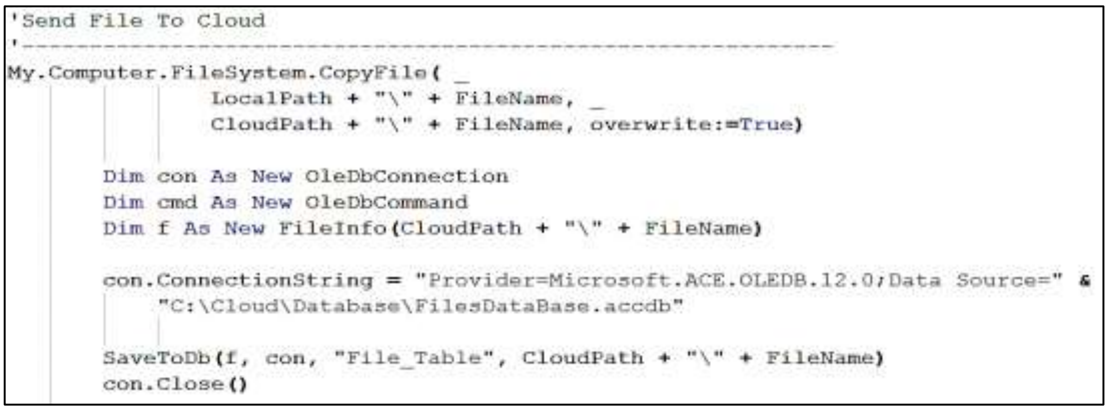


```
'Hasing using sha256
'----------------------------------------------------------
 sha256 = New SHA256Managed()
 Dim hash22 As Byte() = sha256.ComputeHash(memStream1temp)
 memStream1temp.Flush()

'write meta data at the end of the file
'----------------------------------------------------------
     Using writer As StreamWriter = New StreamWriter(fInfo.FullName, True)
        writer.Write(HashedValue)
    End Using
'
```

Figure 4.8: A Snapshot of Hashing the Collected Bytes Using SHA2-256

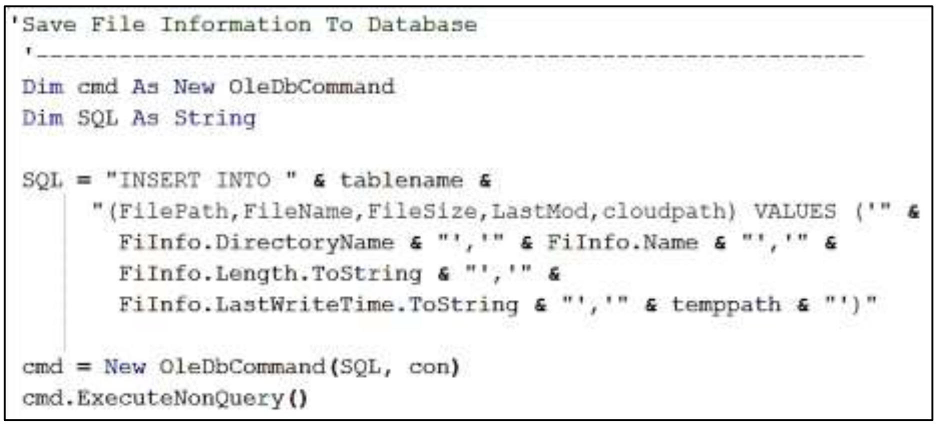
2. Copy to and store class

This class is responsible of send the file to the cloud and save file information to local repository after preparation method is done, the upload speed is depending on the available network bandwidth. In Figure 4.9, we can see how to copy file from local path to cloud path, then save the file information to local database, using SaveToDb function in Figure 4.10.



```
'Send File To Cloud
'------------------------------------------------------------
My.Computer.FileSystem.CopyFile( _
               LocalPath + "\" + FileName, _
               CloudPath + "\" + FileName, overwrite:=True)

       Dim con As New OleDbConnection
       Dim cmd As New OleDbCommand
       Dim f As New FileInfo(CloudPath + "\" + FileName)

       con.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" &
           "C:\Cloud\Database\FilesDataBase.accdb"

       SaveToDb(f, con, "File_Table", CloudPath + "\" + FileName)
       con.Close()
```

Figure 4.9: A Snapshot of Send the File to Cloud Class



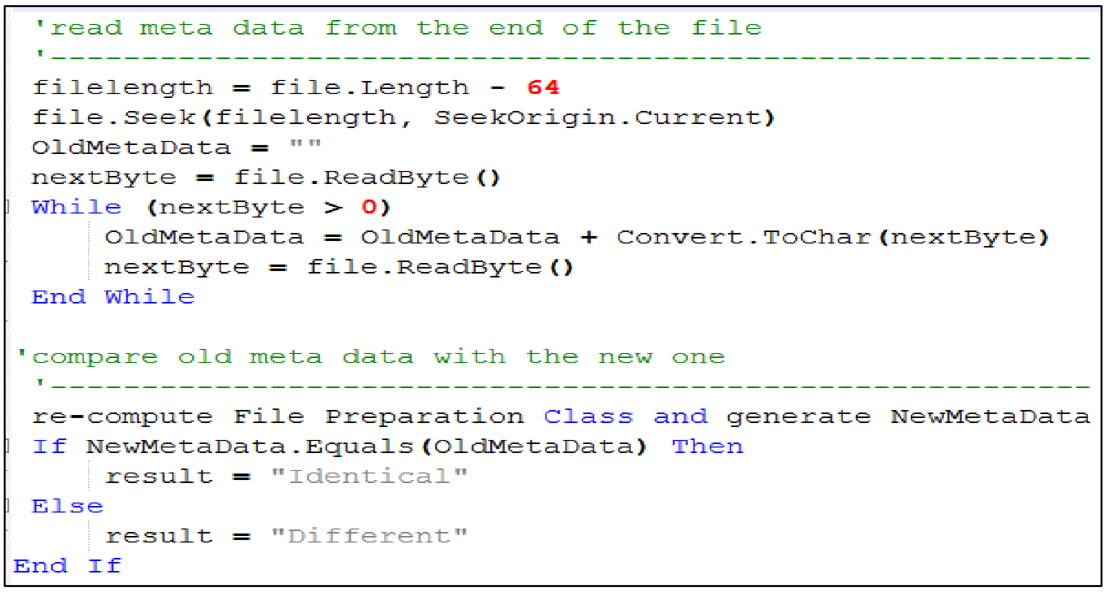
```
'Save File Information To Database
 '------------------------------------------------------------
Dim cmd As New OleDbCommand
Dim SQL As String

SQL = "INSERT INTO " & tablename &
    "(FilePath,FileName,FileSize,LastMod,cloudpath) VALUES ('" &
      FiInfo.DirectoryName & "','" & FiInfo.Name & "','" &
      FiInfo.Length.ToString & "','" &
      FiInfo.LastWriteTime.ToString & "','" & temppath & "')"

cmd = New OleDbCommand(SQL, con)
cmd.ExecuteNonQuery()
```

Figure 4.10: Save File Information into a Database

3. Checksum verifier class

This class is responsible for verifying the file after storing it in the cloud storage to detect any unauthorized modifications.

We accomplished the verifying by re-generating new meta-data from the file without stored hash value, then we compared the old value with the new one. If it matches then the file integrity is saved, otherwise the file integrity is breached.

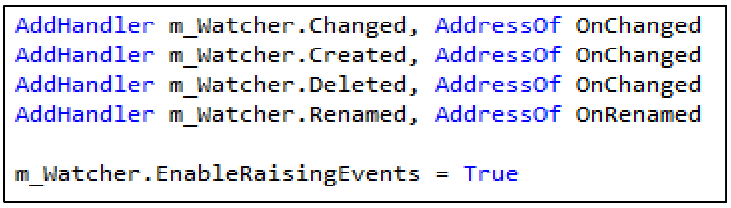


```
 'read meta data from the end of the file
 '---------------------------------------------------------
 filelength = file.Length - 64
 file.Seek(filelength, SeekOrigin.Current)
 OldMetaData = ""
 nextByte = file.ReadByte()
 While (nextByte > 0)
     OldMetaData = OldMetaData + Convert.ToChar(nextByte)
     nextByte = file.ReadByte()
 End While

'compare old meta data with the new one
 '---------------------------------------------------------
 re-compute File Preparation Class and generate NewMetaData
 If NewMetaData.Equals(OldMetaData) Then
     result = "Identical"
 Else
     result = "Different"
End If
```

Figure 4.11: Checksum Verification Class

4. Online auditing class

Listens online to the file system changes notifications and raises events. We can watch changes in files and subdirectories of the specified directory. You can create a component to watch files on a local computer, a network drive, or a remote computer [18].



```
AddHandler m_Watcher.Changed, AddressOf OnChanged
AddHandler m_Watcher.Created, AddressOf OnChanged
AddHandler m_Watcher.Deleted, AddressOf OnChanged
AddHandler m_Watcher.Renamed, AddressOf OnRenamed

m_Watcher.EnableRaisingEvents = True
```

Figure 4.12: Online Auditing Main Events Handler

It is important to mention that this class can watch any changing in a single file or single folder each time

5. **Offline verification**

This class works with local database repository, it compares the file attributes with file information which is stored in local database repository, this verification is very important to detect especially missed files. This database consists of two tables, Files Table and Temp Table.

We used Microsoft access, built a database and created the tables. In Figure 4.13, we can see the structure of file table, the same as in temp table.

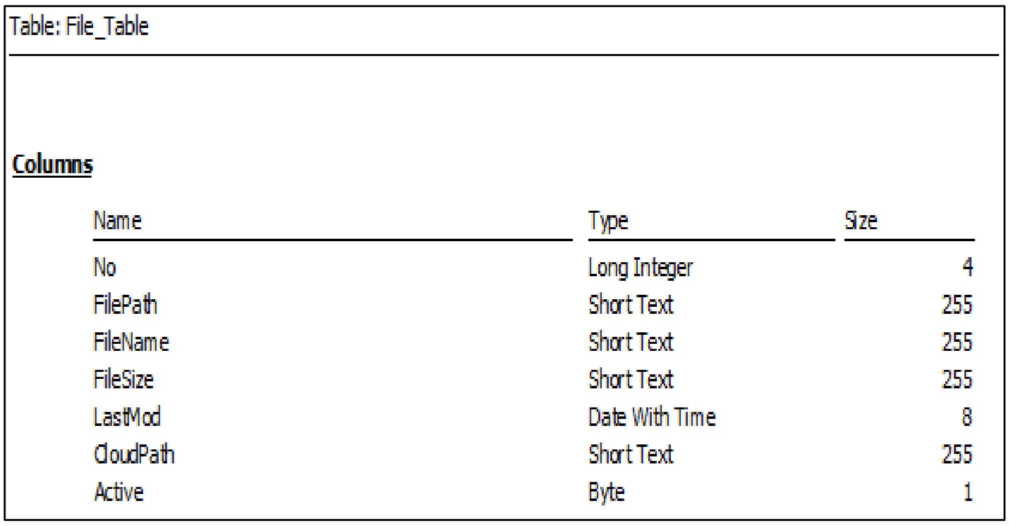


Table: File_Table

Columns

Name	Type	Size
No	Long Integer	4
FilePath	Short Text	255
FileName	Short Text	255
FileSize	Short Text	255
LastMod	Date With Time	8
CloudPath	Short Text	255
Active	Byte	1

Figure 4.13: File Table Structure

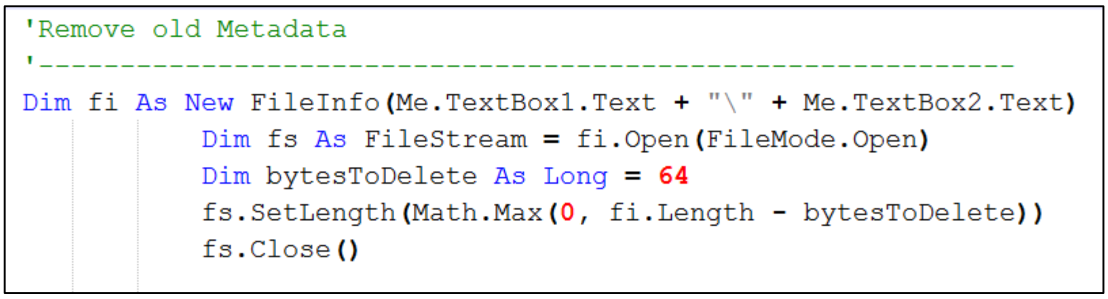
- **File table**: is responsible of storing files information during the process of sending the data file to the cloud storage, as we can see in Figure 4.13 above, the main columns of any File.
- **Temp table**: this table contains the same columns as in file table and is required for verification process, it is better to read all files information from cloud side and store this information into temporary table, after that we can match the two tables, we can detect these cases:
  - ✓ Missing files.
  - ✓ Changing in file size.
  - ✓ Changing in last modify date of the file.

This class is very important when we go online after offline period.

6. Dynamic operations class

The user may need to perform some operations on his data dynamically, it is not necessary to download whole file to his side, he can operate on his own data by simply:

- ✓ Verify the file first.
- ✓ Remove the old hash from the end of the file.
- ✓ Handle some data operations.
- ✓ Regenerate and append the new metadata at the end of the file directly in the cloud.



```
'Remove old Metadata
'------------------------------------------------------------
Dim fi As New FileInfo(Me.TextBox1.Text + "\" + Me.TextBox2.Text)
           Dim fs As FileStream = fi.Open(FileMode.Open)
           Dim bytesToDelete As Long = 64
           fs.SetLength(Math.Max(0, fi.Length - bytesToDelete))
           fs.Close()
```

Figure 4.14: Remove Old Meta-Data From the File

4.2.2 Integrity verifier GUI

As we mentioned before, we built an application using VB.NET, so we design a GUI application for the user, so the user can process and run each class, this Interface contains these components:

1. Local path and Cloud Path: the user must determine the local File path and the cloud file path in order to accomplish an experiment.
2. The GUI contains a button to run every class separately.
3. Present results for each process, we can illustrate the results directly inside the application.
4. Calculate time span in milliseconds for each process.
5. Export results from each process to txt files, this is useful for the evaluation, especially if we work with many files in the same process.

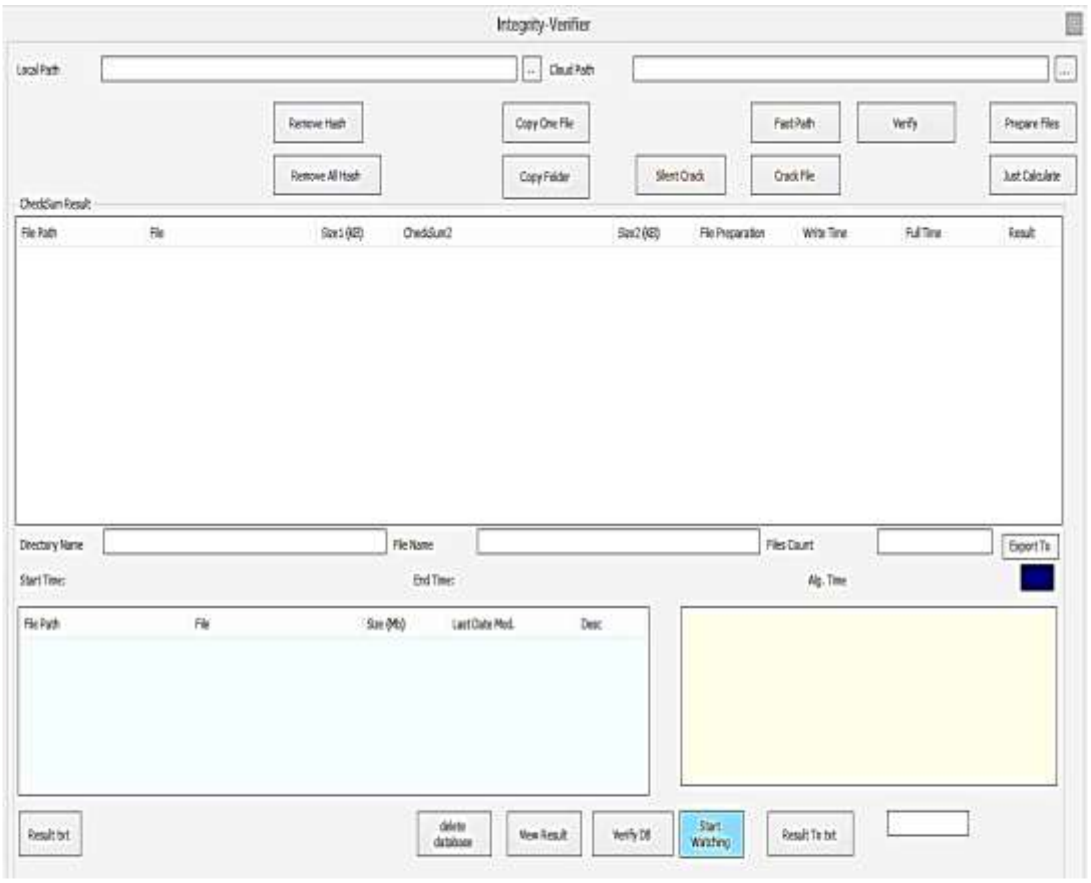


Figure 4.15: Integrity-Verifier Tool GUI

Summary

In this chapter, we presented and built our data integrity model to ensure the Integrity of User's Data in the Cloud. In implementation section, we described the tool which we built, programing language, main classes.

Integrity-Verifier tool, is an application built in VB.NET, it contains 6 classes, we described the main features of each class and how it works.

Chapter 5
Experiments and Evaluations

In this chapter, we present our experiments, and the evaluation of the Integrity-Verifier tool, to ensure the integrity of user's data in the cloud using checksum verification, and detect any unauthorized modifications or missing files.

5.1 Experiments

We performed some experiments on our implementation to demonstrate its ability to verify data integrity, we performed different experiments, which can be listed as:

1. Using different file sizes.
2. Test all methods locally.
3. Verify data integrity on a high speed network and limited network bandwidth with virtual cloud.
4. Using a tool to change some bytes in any file (hack tool).

5.1.1 Dataset

To accomplish our experiment, we must first collect a dataset, our dataset must contain different file sizes and types, so we will be able to collect large amount of data files with 9.16 GB.

Table 5.1: Dataset Description

Size in GB	**Count**	**Size(Byte)**
9.16	1917	9,836,768,217

Some types of data files used are: .doc, .txt, .exe, .pdf, .dll, .wav, .mp3, .mp4, .xls, .dat, .ppt, .bin…etc.

Also, we choose different sizes starting with 11 KB, ending with 1 GB. We grouped it as follows in Table 5.2:

Table 5.2: Dataset Grouping Descreption

Group (File size)	**Count**	**Size(Byte)**
Small (<0.5) MB	819	95,102,961
Medium (0.5-5) MB	764	1,447,213,183
Large (>5) MB	317	2,668,177,290
Huge (>128) MB	17	5,626,274,783

This size grouping is according to our experiment resources, we used windows explorer grouping as an initial values, and then we re-classify this grouping. In Microsoft windows operating system in file explorer, this classification is presented when we want to group files according to its size.

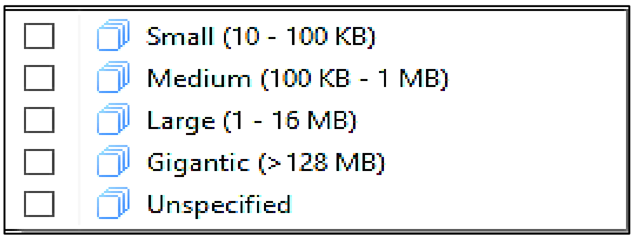

☐ Small (10 - 100 KB)
☐ Medium (100 KB - 1 MB)
☐ Large (1 - 16 MB)
☐ Gigantic (> 128 MB)
☐ Unspecified

Figure 5.1: Microsoft Windows File Size Classifications

We can see in Figure 5.1 that small files are 100KB for max size, in our research we declare that it is important to increase the number of collected bytes as much as possible, to increase the probability of detect any modifications on file.

According to our experiment resources, we grouped the file sizes as mentioned in Table 5.2.

Four subsets of whole dataset extracted according to its size, the size of each of them is 43% small files, 39% medium files, 17% large files and finally 1% huge file.

We cannot collect large number of huge files due to its size, it needs large space, and the collected dataset is suitable.

5.1.2 Experimental environment and tools

Our experiments are conducted with a desktop pc, running windows 7 x86 operating system with core i5, 2400 CPU, @ 3.10 GHz, with 4 GB memory RAM. On VMware Workstation Ver. 10 environment, we created a desktop pc, running windows 7 x86 operating system with one CPU, and with 1 GB memory RAM. We use a free tool to create a private cloud, and web server, called OwnCloud and WampServer.

To test network speed, we use a free tool called, Lan Speed Test.

5.1.3 Setup a private cloud

In this subsection, we will explain the process of setup a private cloud server, we use OwnCloud, and WampServer.

5.1.3.1 WampServer

WampServer is a Windows web development environment. It allows us to create web applications with Apache2, PHP and a MySQL database [34].

5.1.3.2 OwnCloud

OwnCloud declared in its web site as an open source file sync and shared software for everyone, OwnCloud provides a safe, secure, and file synchronization and sharing on servers that we control [21]. We can share files and folders on our computer, and synchronize them with our OwnCloud server.

Place files in our local shared directories, and those files are synchronized to the server and to other devices using OwnCloud Desktop Client. After all, with OwnCloud, it's Your Cloud, Your Data, Your Way [21].

We setup these tools with the help of a video in YouTube, called How to Install OwnCloud on Windows 7 using Wamp [40]. Then we configure OwnCloud with WampServer, create admin user and test user, finally, we download and install the desktop client tool of OwnCloud, it is look like famous cloud storage providers, google drive and dropbox.

5.1.4 Model experiments

After we collected suitable data set, and setup a private cloud server, seven different types of experiments have been constructed.

- **Experiment 1**: Preparing and verifying files locally.
- **Experiment 2**: Verify files in the cloud using high speed network.
- **Experiment 3**: Verify files in the cloud using limited network bandwidth.
- **Experiment 4**: Work on one file in the cloud using limited network bandwidth.
- **Experiment 5**: Online auditing
- **Experiment 6**: Offline verification
- **Experiment 7**: Verify files in public cloud "Dropbox".

5.1.4.1 Experiment 1: Prepare and verify files locally

The first experiment is to prepare files locally and append meta-data to each file, by using different file sizes. The results occur as in Table 5.3, it is grouped according to file size:

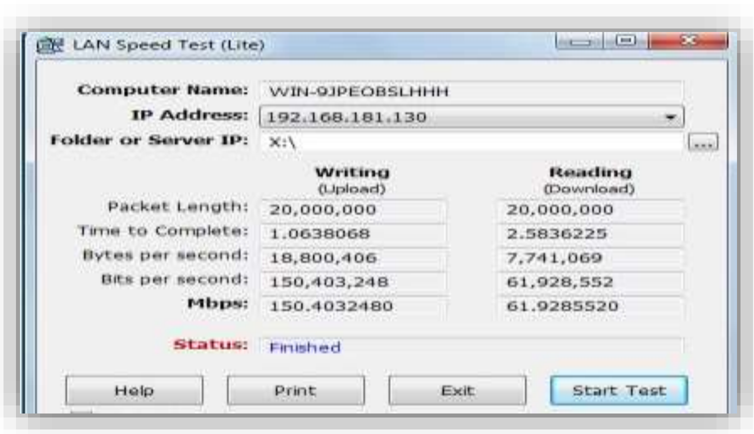
Table 5.3: Prepare and Verify Files Locally Results

Desc	Small	Medium	large	Huge
Files Count	819	764	317	17
Files size, MB or GB	90.6 MB	1.34 GB	2.48 GB	5.23 GB
Files size(Byte)	95,102,961	1,447,213,183	2,668,177,290	5,626,274,784
Prepare Files(Ms)	11,250	36,400	36,362	7,707
Verify Files(Ms)	10,361	2,382	624	32
Save To Data Base	2,277	2,373	343	649

We prepared 1917 file locally using a desktop pc with suitable performance, with different file sizes and types, it takes less than 2 minutes in preparation, and less than 15 seconds for verification.

5.1.4.2 Experiment 2: Verify files in the cloud using high speed network

After sending files to the cloud, and to accomplish our main objective in this research, we need to verify the data in the cloud side. A user maybe an organization, which has high resources and high speed network, or a normal user with limited resources, so in order to evaluate our model, we conduct an experiment for each situation. This experiment is conducted with high speed network, the network speed was:


LAN Speed Test (Lite)

Computer Name: WIN-9JPEOBSLHHH
IP Address: 192.168.181.130
Folder or Server IP: X:\

	Writing (Upload)	Reading (Download)
Packet Length:	20,000,000	20,000,000
Time to Complete:	1.0638068	2.5836225
Bytes per second:	18,800,406	7,741,069
Bits per second:	150,403,248	61,928,552
Mbps:	150.4032480	61.9285520

Status: Finished

Figure 5.2: Network High Speed Measures

Download speed: 61 Mbps, **upload speed:** 150 Mbps.

The results of Experiment 2 is:

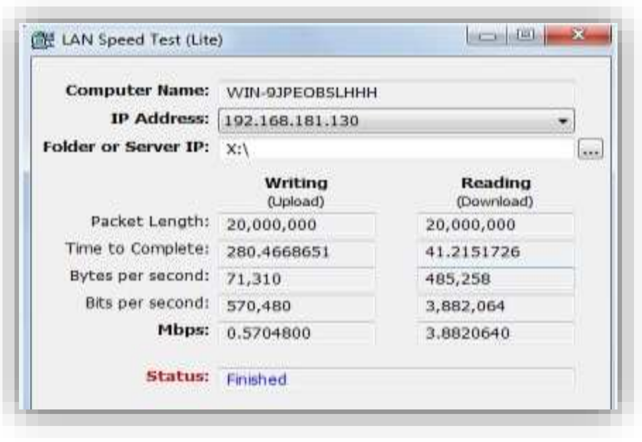
Table 5.4: Verify Files in the Cloud Using High Speed Network Results

Desc	Small	Medium	large	Huge
Files Count	*819*	*764*	*317*	*17*
Files size, MB or GB	*90.6 MB*	*1.34 GB*	*2.48 GB*	*5.23 GB*
Files size(Byte)	*95,102,961*	*1,447,213,183*	*2,668,177,290*	*5,626,274,784*
Verification Files				
Iteration 1	15,250	69,018	68,922	10,453
Iteration 2	7,346	48,013	8,601	64
Iteration 3	6.711	15,403	3,662	48
Iteration 4	5,544	11,963	862	32
Iteration 5	5,226	10,454	752	-

In this experiment, we verify all files in the cloud in less than 3 minutes at worst case which was appeared in iteration 1. We think that the gap between first time span of first iteration and the last one appears from the problem of First Byte Time, the time to First Byte is the amount of time it takes after the client sends a request to receive the first byte of the resource from the server.

5.1.4.3 Experiment 3: Verify files in the cloud using limited network bandwidth

As we mentioned above, a user maybe an organization, or a normal user with limited resources, this experiment is conducted with the same desktop pc, and network speed, which was:


LAN Speed Test (Lite)

	Writing (Upload)	Reading (Download)
Computer Name:	WIN-9JPEOB5LHHH	
IP Address:	192.168.181.130	
Folder or Server IP:	X:\	
Packet Length:	20,000,000	20,000,000
Time to Complete:	280.4668651	41.2151726
Bytes per second:	71,310	485,258
Bits per second:	570,480	3,882,064
Mbps:	0.5704800	3.8820640
Status:	Finished	

Figure 5.3: Network Limited Speed Measures

Download speed: 3.8 Mbps, **upload speed:** 0.57 Mbps.

The results of Experiment 3 is:

Table 5.5: Verify Files in the Cloud Using Limited Network Bandwidth Results

Desc	Small files	Medium files	large files	Huge files
Files Count	*819*	*764*	*317*	*17*
Files size, MB or GB	*90.6 MB*	*1.34 GB*	*2.48 GB*	*5.23 GB*
Files size(Byte)	*95,102,961*	*1,447,213,183*	*2,668,177,290*	*5,626,274,784*
Verification Files				
Iteration 1	162,029	383,428	174,027	10,947
Iteration 2	158,239	380,724	173,795	10,983

In this experiment, we verify all files in the cloud in less than 13 minutes, the worst case appeared in mostly iteration one.

5.1.4.4 Experiment 4: Dynamic operation support in the cloud using limited network bandwidth

We conducted also an experiment in one file for each group of file size, this experiment is required to test dynamic data operations, when we need to remove the hash value from the file, which is stored in the cloud.

The results of experiment four was:

Table 5.6: Dynamic Operation Support Using Limited Network Bandwidth Results

Desc	Small	Medium	Large	Huge
Files Count	*1*	*1*	*1*	*1*
Files size(Byte)	*155,649*	*5,011,766*	*16,317794*	*1,085,828,878*
Collected Bytes	155,649	1287	1472	2312
Verify Files(ms)	203	452	499	671
Remove Hash (ms)	4	5	6	7
Prepare Files(ms)	187	437	486	640

In this experiment, we verify one files in the cloud in less than 0.5 second for most files.

5.1.4.5 Experiment 5: Online auditing:

In this research, we introduce a new idea to ensure the integrity of user's data in the cloud which is, online auditing. We used a method called File System Watcher Class, which listens to any change of a directory, or file in a directory.

We tested this class, and we found that it is very efficient to detect any change if the it raises in file size or last date modification of the file, there are very important attributes that we can use to detect any change in any file,

We try to expand its features to detect more than one file at the same time, so to detect more than one file we try to increase the internal buffer size in bytes. (The default is 8192 (8 KB).)
The buffer starts from 4 KB, but it must not exceed 64 KB. For best performance, use a multiple of 4 KB on Intel-based computers. To declare the Internal Buffer Size, we quote the declaration from Microsoft:

"*The system notifies the component of file changes, and it stores those changes in a buffer the component creates and passes to the APIs. Each event can use up to 16 bytes of memory, not including the file name. If there are many changes in a short time, the buffer can overflow. This causes the component to lose track of changes in the directory, and it will only provide blanket notification. Increasing the size of the buffer can prevent missing file system change events." [18]*.

So, we choose the buffer size=16 kb (m_Watcher.InternalBufferSize = 16384). After we change the buffer size, we can detect more than one file at the same time.

5.1.4.6 Experiment 6: Offline verification using limited network bandwidth

Offline verification, is another verification method which introduces in our model, it aims to provide the user the ability to scan his files quickly, in order to detect any missed files or detect any modifications in his data files, we conducted this experiment using our database repository, it is done by scan all files and store files attributes in temp table, and compare scanned files with old values in file table. This experiment is conducted in limited network bandwidth.

In Table 5.7, we introduced our experiment calculations:

Table 5.7: Offline Verification Using Limited Network Bandwidth

Desc	**Small**	**Medium**	**large**	**Huge**
Files Count	*819*	*764*	*317*	*17*
Verify Files(ms)	1613	7771	2551	778
Avg. of one File (ms)	1613/819 ≈ **1.97**	7771/764 ≈ **10.17**	2551/317 ≈ **8.05**	778/17 ≈ **45.76**

In this experiment, we verify all files (1917 file) in the cloud in less than 13 seconds.

5.1.4.7 Experiment 7: Verify files in real public cloud "Dropbox"

It is important in our opinion, to run an experiment in public cloud, in order to ensure that our model is suitable and performed well, so we execute our model in Dropbox cloud provider.

Dropbox

"Dropbox is a service that allows you to bring all your photos, docs, and videos anywhere, and share them easily. Any file you save to your Dropbox will automatically be saved to all your computers, your phone or tablet, and the Dropbox website. Dropbox also makes it easy to share with others. If your computer melts down, you can restore all your files from the Dropbox website with a couple of clicks", Dropbox has more than 300,000,000 users, people save 1 billion files to Dropbox every 24 hours [9].

We conducted this experiment in Dropbox for all methods we presented, starting from collecting a dataset of 99 files, with different sizes from small to medium files. It is not possible for us to upload a huge number of files, because of our available uploading speed, it is not greater than 60 Kbps, total file size is (53.4 MB), and the results are shown in Table 5.8, as follows:

Table 5.8: Results of Verifying Files in "Dropbox"

Desc.	**Different files size**
Files Count	*99*
Files size(MB)	*53.4*
Files size(Byte)	56,037,955
Prepare Files(ms)	34,863
Verify Files(ms)	28,040
Remove Hash (ms)	1,115
Save to Database	1427
Verify from Database	832

In this experiment we also tested online auditing, since dropbox provides a way to access our files using mobiles, we use our mobile to delete and modify some files, and the system catches these operations.

5.2 Evaluation of the model

The most important measure is performance, it depends on the execution time; which is related to our algorithm of verification of large files.
The other measure is to evaluate the efficiency in terms of security, it is measured by detecting if any files are deleted or modified either online or offline.

5.3 Testing environment

A testing environment is a setup of software and hardware on which the testing is performed. This setup consists of the physical setup which includes hardware, and logical setup that includes Server Operating system, client operating system, database, front end running environment

So we accomplished that by:

1- Setup a virtual private cloud.
2- Setup a client pc, using VMware Workstation V10, this pc running windows 7 x86 operating system.
3- Collect suitable dataset, collect 1917 files with different sizes, and group these files into four categories.
4- Install our tool "Integrity Verifier", to client pc, as an application, then integrate the tool with local database.
5- Integrity verifier tool contains main methods, checksum verification on each file, online auditing and offline verification.

5.4 Performance evaluation

Several experiments are conducted, in order to evaluate the performance depending on execution time aspect.

5.4.1 Checksum verification performance

This measure is very important, especially with work in large files in the cloud environment, as we seen before in Section 5.1.4, we can notice clearly that the proposed model is fast, we summarized the results as follows:

Table 5.9: Results Summary of Verification Experiments

Description	Files			Time (ms)
	count	size	group	
Local verifications	819	90.6 MB	small	11,250
High network speed				15,250
Limited Network				162,029
Local verifications	764	1.34 GB	Medium	36,400
High network speed				69,018
Limited Network				383,428
Local verifications	317	2.48 GB	Large	36,362
High network speed				68,922
Limited Network				174,027
Local verifications	17	5.23 GB	Huge	7,707
High network speed				10,453
Limited Network				10,947
Public Cloud	99	53.4 MB	Variant	28,040
Working with One File				
Limited Network	1	152 KB	Small	203
	1	5 MB	Medium	452
	1	15 MB	Large	499
	1	1 GB	Huge	671

According to the summarization presented in Table 5.9, we can notice that the performance when we run all tests in limited network bandwidth is not the least time, but it is logical result, and we can extract some observations from it as follows:

Table 5.10: Calculate the Average Time for One File from Verification Experiment

Description	Files			Total time (ms)	Avg. Time
	count	size	Avg.		
Limited Network bandwidth	819	90.6 MB	90.6/819 ≈ **0.11 Mb**	162,029	162.029/819 ≈ **198 ms**
	764	1.34 GB	1.34/764 ≈ **1.79 MB**	383,428	383,428/764 ≈ **501 ms**
	317	2.48 GB	2.48/317 ≈ **8 MB**	174,027	174,027/317 ≈ **549 ms**
	17	5.23 GB	5.23/17 ≈ **315 MB**	10,947	10,947/17 ≈ **644 ms**

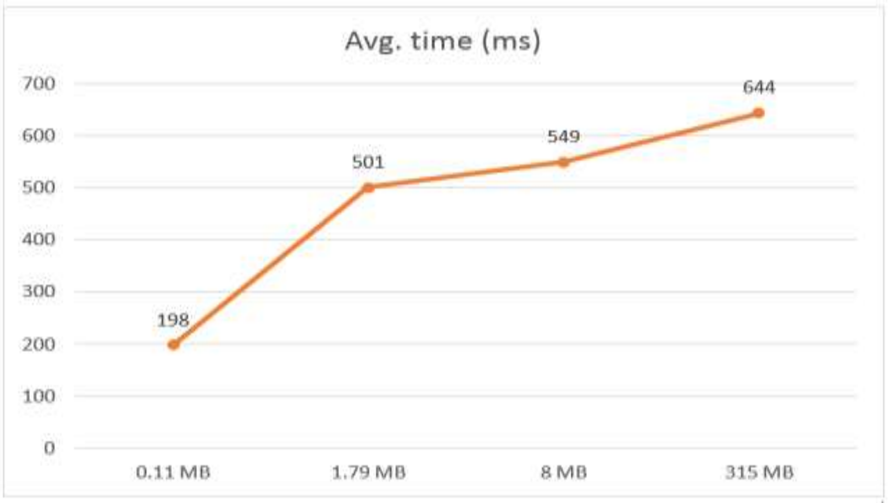


Figure 5.4: Average Time for One File from Verification Experiment graph

We can say that each file with size greater than 0.5 MB need approximately 0.5 second to verify. This is because of retrieving data from file for H several times, hits the file from the cloud server several times, we can declare this in Table 5.11:

Table 5.11: Time Needed to Read from File F for H Hit Number

Size(MB)	Size (Byte)	H (Hits count)	time	Hit time(ms)
1.79	1876951	63	501	7.95
8	8388608	75	549	7.32
315	330301440	84	644	7.66

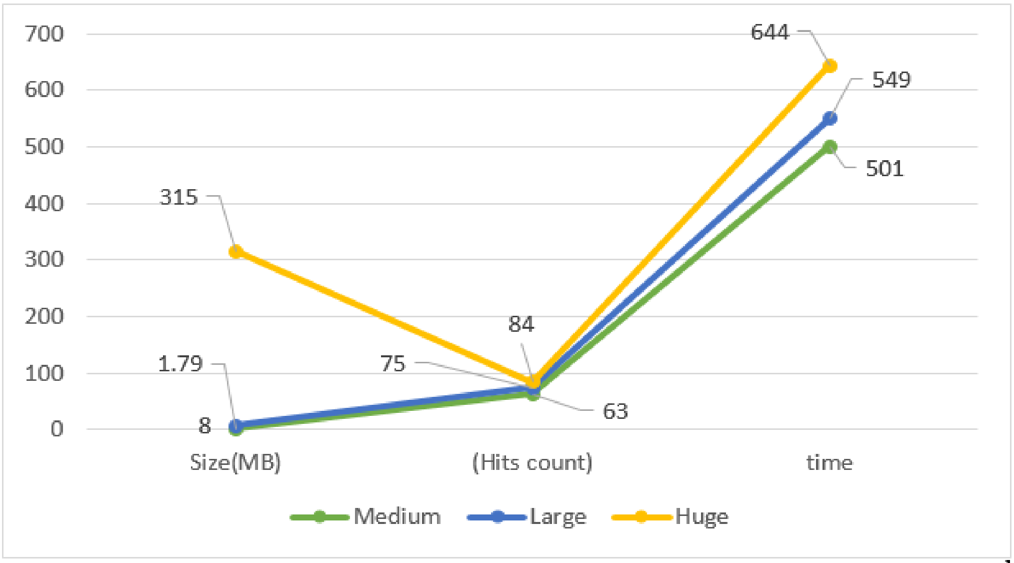


Figure 5.5: Time Needed to Read from File F for H Hit Number Graph

We can extract hit count value approximately from ((n+m)*3), hit time value: is the time to complete reading of L byte, which block size=2^L.

We can see that we Collected a reasonable number of bytes with respect to the size of data file (in bytes), as we mentioned before, it is better to collect large number of bytes as possible, this is necessary to increase the probability of detection of any unauthorized modifications, also, network bandwidth must minimized, as the size of the proof is comparatively very small according to the file size.

In this table we must take in our consideration that it is very efficient to minimize number of hits to each file, also try to collect much bytes as possible from different positions in the file, this method increased the probability to detect the changes in the file.

5.4.2 Offline verification

In our model we introduced this verification in order to ensure that our files are not missing, especially when a user disconnect his connectivity with internet for any reason, or if he want to store some family pictures for years as an example, so the main purpose of this verification method is to detect missing files, we found also that this method is the fastest method in our model, as we can see in Table 5.7, the verification time is less than 13 second for all data files in our dataset which contains 1917 file, we recommend to use this verification, when the user connect again to the internet.

5.4.3 Online auditing

Online auditing is a watcher on single directory or a single file inside a directory, this method uses internal buffer, which must not exceed 64 KB. We used 16 KB in our experiment in order to increase the probability to detect any changes in more than one file at the same time. Therefore, the size of received amount of data from cloud server, if any change occurs, is not large. When we monitor the network bandwidth, we found that the listener needs less than 1KB every second, and when detecting any changes it still less than 16 KB in a second, we found that, the received amount of data is very acceptable, and does not affect the network bandwidth.

5.4.4 Dynamic operation support

The most important issue here is to conduct the change in the server side without the need to download all data locally, the process to remove the old hash value, recomputed a new one and appended it at the end of the file takes reasonable time as we can see in Figure 5.6.

Table 5.12: Update Operation in Cloud Side

Desc	Size	Verify	Remove Hash	Prepare File	Total
Small	*0.15 MB*	*203*	*4*	*187*	*394*
Medium	*5 MB*	*452*	*5*	*437*	*894*
large	16 MB	499	6	486	991
Huge	1 GB	671	7	640	1318

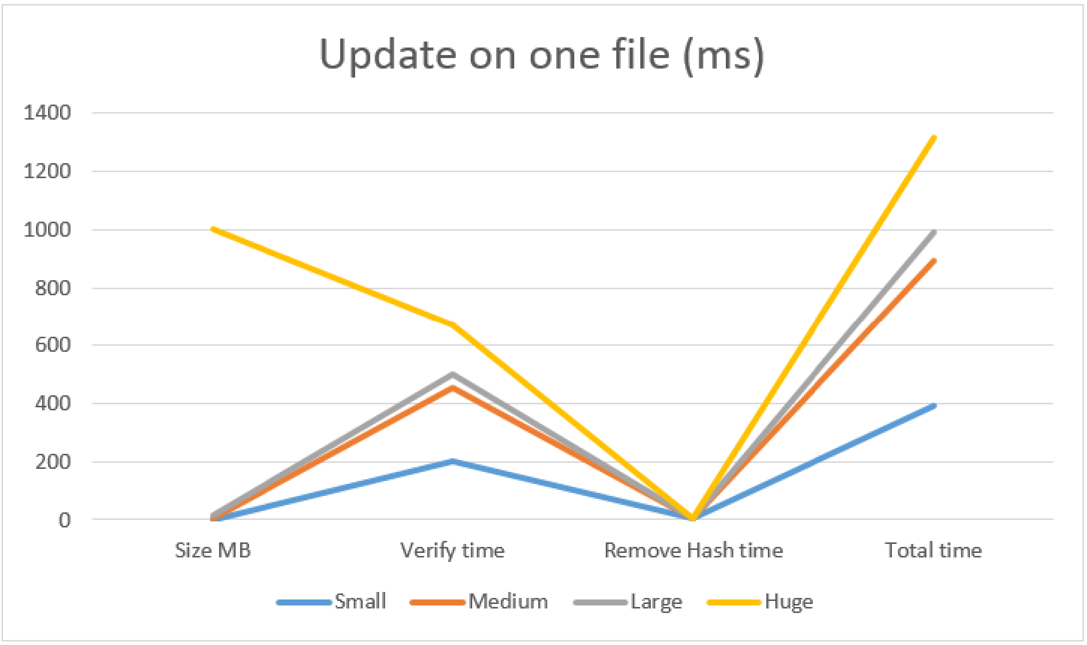


Figure 5.6: Update Operation in Cloud Side Graph

5.5 Efficiency evaluation

Efficiency measures the output results, and in our model we evaluate the efficiency in term of security. A security evaluation proves that the functions have been implemented based on our requirements, and the output results of this measure proves that our model has been performed at an agreed level. Different types of tests were conducted to ensure our works.

Our security analysis focuses on detection data modification or deletion at server side we modify and delete some data files randomly, the modification is done by several way like direct modification or modifying and deleting other bytes in different files, we can read in verifications results that our model is highly efficient and resilient malicious data modification attacks.

5.5.1 File verification

The most important method in our research, is how to ensure the integrity of user's data in the cloud, with respect to file size and network bandwidth, we conducted many tests in order to evaluate the efficiency of this method, and these tests included:

1- Insert some bytes to the file.

2- Delete some bytes from the file.

3- Change some bytes in the file, by using a free tool to crack some files in the cloud.

4- Update some files directly in the server side.

5- Silent crack, means to crack any file and saving the size and last modification date of the file.

We choose randomly some files from our dataset with these count, in Table 5.13:

Table 5.13: Count of Selected Files to Crack

Desc.	**Files Count C**	**C1=70% Of C**	**C2=30% Of C**	**C3=30% from C1**	**Cracked Files = 30% of C3**
Small	819	573	246	172	51
Medium	764	534	230	161	48
Large	317	221	96	67	20
Huge	17	11	6	4	1
Total	**1917**	**1341**	**576**	**403**	**120**

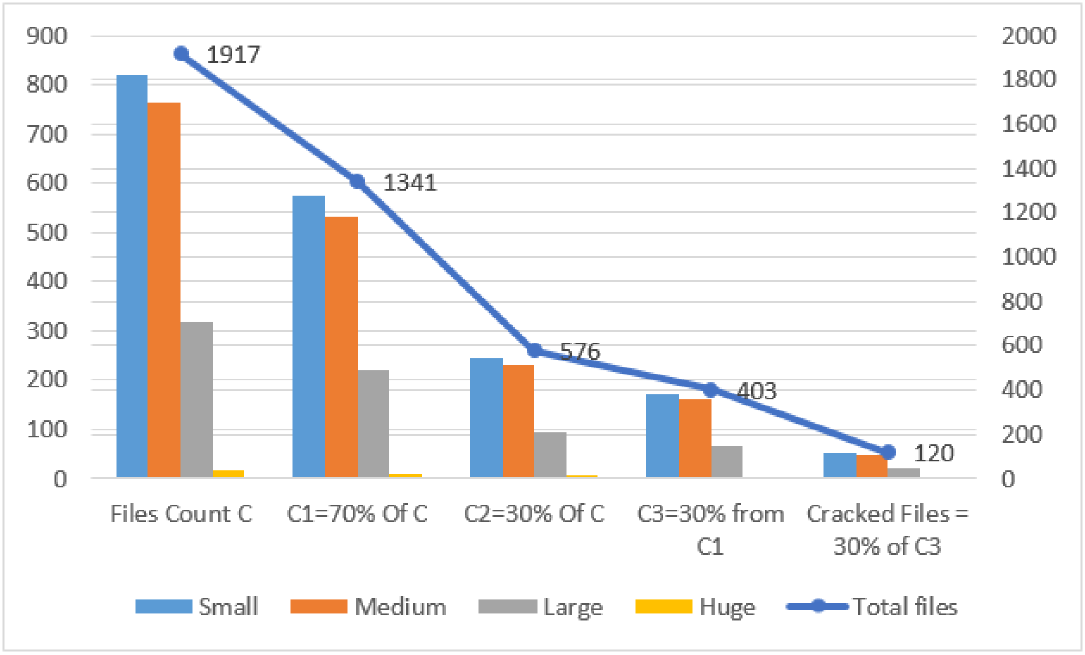


Figure 5.7: Count of Selected Files to Crack Graph

After we conducted all these tests, we verified our dataset, the system was very effective and detected all files that were modified.

This is done by re-computing our method and recollecting the bytes from the file and calculating a hash value after inserting the secret key which is known only by the user, and comparing the hash value with old hash value which is stored in the file as a meta-data. If there is any mismatches between hashes values, it will give us a result that the file is different from the original one. The system will produce the new hash value and the recovered hash value to the user.

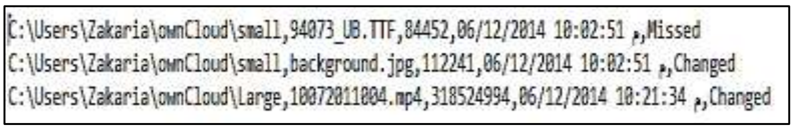
This method cannot detect any missing files, this case can be detected on offline verification step only.

Collision resistant

We checked our dataset SHA256 hash value for all 1917 files, after we prepared the files. We found that SHA256 is resistant to collision attacks, and we did not find any two different files having the same hash value whether the file was small or large.

5.5.2 Offline verification results

When a user reconnect to the internet, he can perform this verification in order to verify that his data is not modified or deleted. We found in Subsection 5.4.2 that this method is the fastest method. So we recommended that this is the first method that should run in our data verification methods. In this verification process, if there is any result, the system will introduce these information:



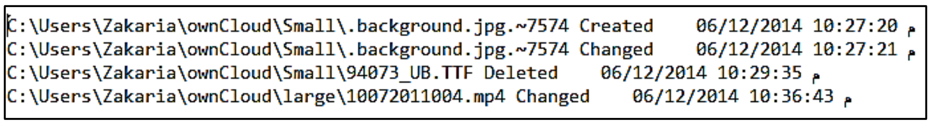
```
C:\Users\Zakaria\ownCloud\small,94073_UB.TTF,84452,06/12/2014 10:02:51 م,Missed
C:\Users\Zakaria\ownCloud\small,background.jpg,112241,06/12/2014 10:02:51 م,Changed
C:\Users\Zakaria\ownCloud\Large,10072011004.mp4,318524994,06/12/2014 10:21:34 م,Changed
```

Figure 5.8: Offline Verification Results

It is important to say that this method is very useful to detect missed files, we conducted many tests in order to test this method. It is very effective, but this method cannot detect only the case of modifying files without changing the size or last modification date in file attributes, this is not difficult to any cracker or cloud provider.

5.5.3 Online auditing result

File System Watcher will provide these results if any modification or deleting is detected in main or sub folder.



```
C:\Users\Zakaria\ownCloud\Small\.background.jpg.~7574 Created    06/12/2014 10:27:20 م
C:\Users\Zakaria\ownCloud\Small\.background.jpg.~7574 Changed    06/12/2014 10:27:21 م
C:\Users\Zakaria\ownCloud\Small\94073_UB.TTF Deleted    06/12/2014 10:29:35 م
C:\Users\Zakaria\ownCloud\large\10072011004.mp4 Changed    06/12/2014 10:36:43 م
```

Figure 5.9: Online Monitor Result

Also, we tested this method in dropbox public cloud, we started the watcher on our pc, and deleted some files from my mobile, which can access to my data files in dropbox, and it works nicely and the system detect these modifications. Also, this method can detect the changing or deleting of more than one file at the same time, but it still limited it can detect only from 5 to 20 files only at the same time, We discussed this issue and we argue that it is still very important method to detect any unauthorized modification online, since we try to be online all the time. Therefore if any unauthorized modification is detected, we can work with a response plan to prevent our files to lose or malicious modification to other files.

Summary

In this chapter, seven different types of experiments have been conducted to demonstrate the ability to verify data integrity. We performed these experiments using different file sizes, test locally and over a network, and with different bandwidth.

These experiments covered all possible situations, in a high speed network and limited one, also covered all file sizes and types, which we considered as an important contribution in this thesis.

We noted the results for each experiment in order to evaluate and discuss these results.

In evaluation section, we evaluated our model in terms of performance and efficiency. We found that any file verification process of large files needs about 0.5 seconds according to limited network bandwidth.

It is necessary to say that, the File Verification process is the most important method in this research. We proved that it is very effective, it detected all modified files, with high performance. Also, Offline verification is the fastest method in our model, it is very effective especially on detecting the missing files.

Chapter 6
Conclusion and Future Work

In this chapter, we concluded our work, results, and the future work directions.

In this research, we discussed the problem of ensuring data integrity in the cloud.

We introduced the security problems in the cloud and declared that data security is becoming a fundamental obstruction in cloud computing, and it is important to users to ensure that their data is stored securely without the existence of local copies.

Many researchers discussed this problem and introduced some solutions to ensure data integrity, which is stored in the cloud.

Most prior works considers static data files and not supporting dynamic data operation or introducing significant computation and communication complexity. None of them discussed the problem of large data files with dynamic operation support; also, they did not study the problem of missing data files.

So, we built the model to ensure the integrity of user's data in the cloud without the need to keep old data files locally in user's side. This model provided three methods:

1- **File verification method**

   It is considered the most important method in our model; we collected a reasonable number of verification bytes depending on file size to increase the probability of detecting any unauthorized modifications.
   This method reduced computational overhead of the client and reduced the network bandwidth.
   Several experiments are conducted, we used different size of files start from 11 KB up to 1 GB. We found that file verification process for each file needs about 0.5 seconds.

   A suitable hash algorithm is used to encrypt the collected bytes with a secret key and then we wrote the hashed value at the end of the file.

We found that the chosen of SHA256 algorithm is a correct choice, it is very effective and collision resistant.

2- **Offline verification**

When we got offline for a while, and reconnected online again. This verification is used to detect any missed files and proved that it is the fastest verification.

This method is fast and effective, it detected all files with a change in size and last modified date, and the best effective feature was detecting all missing files.

In the experiment in limited network bandwidth we conducted this method in less than 13 seconds for all (1917) files.

3- **Online monitoring**

Online auditing is introduced as a new idea, when most users connect to the internet all the time, to follow their social media. We used this feature and introduced online watcher.

We illustrated that this verification is very useful for the user, in order to detect any modifications directly.

Also, dynamic operation support is allowed.

Unlike most prior works, we introduced a real experiments with real results, most of previous researches did not provide their results, and also we tried to cover most scenarios in our seven experiments with our collected large dataset.

We found that our model is very effective and of high performance

Future work

Our model does not prevent the modifications of data. It cannot detect or recover the original data if any unauthorized modification is accomplished.

Also our model does not support other users or third party auditors to verify the data, the model allow only the owner user to verify his data, all these will be a future challenges.

References

[1] "The NIST Definition of Cloud Computing", National Institute of Standards and Technology. September 2011.

[2] Ari Juels and J. Burton S. Kaliski, “PORs: Proofs of Retrievability for Large Files” Proc. of CCS ’07, pp. 584–597, 2007.

[3] *B Anusha, Y. Ramu*, "Storage Correctness and Dynamic Data Support For Cloud Data", International Journal of Research in Computer and Communication Technology, VOL 2, NO 10, 2013. http://www.ijrcct.org/index.php/ojs/article/view/420",

[Accessed on: 30/05/2014]

[4] Chris Marsh, "Data Integrity In The Cloud", MAY 2011. http://www.wwpi.com/~wwpi/index.php?option=com_content&view=article&id=12800:data-integrity-in-the-cloud&catid=99:cover-story&Itemid=2701018, *[accessed on: 30/05/2014]*

[5] Cong Wang, Qian Wang, and Kui Ren, Wenjing Lou, "Ensuring Data Storage Security in Cloud Computing", IEEE, Quality of Service. IWQoS. 17th International Workshop, July 2009

[6] D. Bhu Lakshmi and S. Arundathi, " Providing Privacy and Security for Cloud Data Using Data Mining", International Journal of Innovation and Scientific Research, ISSN 2351-8014 Vol. 11 No. 2, pp. 264-272, Nov. 2014,

[7] "Deprecation of SHA-1 and moving to SHA-2", https://support.servertastic.com/deprecation-of-sha1-and-moving-to-sha2/, *[Accessed on: 30-11-2014]*.

[8] "What is Cloud Computing", https://www.dialogic.com/Solutions/Cloud-Communications/Learn/Overview-of-Cloud-Communications.aspx , *[Accessed on 12-12-2014]*

[9] *"What is Dropbox?",* https://www.dropbox.com/news/company-info, *[Accessed on 11-12-2014].*

[10] https://www.eucalyptus.com/sites/all/img/product/diagram-cloud-computing-characteristics.png , "What is Cloud Computing",

[accessed on: 09/05/2014]

[11] Giuseppe Ateniese, Roberto Di Pietro, Luigi V. Mancini, and Gene Tsudik, “Scalable and Efficient Provable Data Possession,” *Proc. of SecureComm ’08*, pp. 1–10, 2008.

[12] Jan de Muijnck-Hughes, "Data Protection in the Cloud", Master Thesis, Radboud University Nijmegen, Nijmegen, Gelderland, the Netherlands, March 2011.

[13] K.GEETHA, ANANTHI SHESHASAAYEE, "Survey for security issues in the cloud computing data", International Journal of Advances In Computer Science and Cloud Computing, ISSN: 2321-4058 Volume- 1, Issue- 2, Nov-2013.

[14] Kevin D. Bowers, Ari Juels, and Alina Oprea, “Proofs of Retrievability: Theory and Implementation” Cryptology ePrint Archive, Report 2008/175, http://eprint.iacr.org/, 2008

[15] Kui Ren, Cong Wang and Qian Wang, “Security Challenges for the Public Cloud,” IEEE Internet Computing, vol. 16, no. 1, pp. 69-73, 2012.

[16] Kuyoro S. O., Ibikunle F. & Awodele O., " Cloud Computing Security Issues and Challenges", International Journal of Computer Networks (IJCN), Volume (3) : Issue (5) : 2011

[17] Mehul A. Shah, Mary Baker, Jeffrey C. Mogul, Ram Swaminathan, “Auditing to Keep Online Storage Services Honest” Proc. 11th USENIX Workshop on Hot Topics in Operating Systems (HOTOS ’07), pp. 1–6, 2007

[18] Microsoft Developer Network MSDN, "FileSystemWatcher Class", .NET Framework 4.5, http://msdn.microsoft.com/en-us/library/system.io.filesystemwatcher(v=vs.110).aspx,

[Accessed on 06-12-2014].

[19] Miss. M.Sowparnika1, Prof. R. Dheenadayalu2," Improving data integrity on cloud storage services ". International Journal of Engineering Science, ISSN (Print): 2319 – 6726, PP.49-55, February. 2013

[20] Mr. Gurmeet Singh; Mr. Vineet Kumar Sachdeva, "Impact and Challenged of Cloud Computing in current scenario", International Journal of Social Science & Interdisciplinary Research, ISSN 2277 3630, Vol.1 Issue 10, October 2012.

[21] OwnCloud User Manual, http://doc.owncloud.org/server/7.0/user_manual, *[Accessed on 08-12-2014]*

[22] Pravin Rathod, Subhangi Sapkal, "Audit Service for Data Integrity in Cloud", International Journal of Advanced Research in Computer Science and Software Engineering, ISSN: 2277 128X, Volume 4, Issue 4, April 2014

[23] R. Christiaanse, J. Hulstijn, "NEO-CLASSICAL PRINCIPLES FOR INFORMATION INTEGRITY", VU University and EFCO Solutions; Delft University of Technology and Thauris B.V. The Hague, 2011

[24] Randeep Kaur, Supriya Kinger, " Analysis of Security Algorithms in Cloud Computing", International Journal of Application or Innovation in Engineering & Management (IJAIEM), ISSN: 2319 - 4847 Volume 3, Issue 3, March 2014

[25] Salma, T.J., " A Flexible Distributed Storage Integrity Auditing Mechanism in Cloud Computing", International Conference on Information Communication and Embedded Systems, ISBN: 978-1-4673-5786-9 (Print) PP. 283 – 287, February. 2013

[26] Saranya Eswaran and Dr.Sunitha Abburu, "Identifying Data Integrity in the Cloud Storage" IJCSI International Journal of Computer Science Issues, ISSN (Online): 1694-0814, Vol. 9, Issue 2, No 1, March 2012

[27] Selman Haxhijaha, Gazmend Bajrami, Fisnik Prekazi, " Data Integrity Check using Hash Functions in Cloud environment ", International Conference in business, innovation and technology, ISBN: 978-9951-437-24-0, p 113-118,Nov 2013.

[28] Sravan Kumar R, Saxena, A., "Data Integrity Proofs in Cloud Storage", Communication Systems and Networks, Third International Conference, Bangalore, Jan. 2011.

[29] Talib, A.M., Atan, R. ; Abdullah, R. ; Azrifah, M., "CloudZone: Towards an Integrity Layer of Cloud Data Storage Based on Multi Agent System Architecture", IEEE Conference on Open Systems, , Langkawi, Malaysia, PP. 127-132, September. 2011

[30] Thomas Schwarz, S.J. and Ethan L. Miller, “Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage” Proc. of ICDCS ’06, pp. 12–12, 2006.

[31] Trusted booster, SSL expert since 1996, "All about SHA1, SHA2 and SHA256 hash algorithms", https://www.tbs-certificates.co.uk/FAQ/en/sha256.html, *[accessed on: 30-11-2014]*.

[32] Tutorialspoint, Simply Easy Learning, "VB.Net Programming Tutorial", http://www.tutorialspoint.com/vb.net/, *[Accessed on 05-12-2014]*

[33] Vytautas Zapolskas, " Securing Cloud Storage Service ", Master of Science Thesis, Stockholm, Sweden, 2012

[34] WampServer, a windows web development environment, http://www.wampserver.com/en, *[Accessed on 08-12-2014]*

[35] From Wikipedia, "Cloud storage", http://en.wikipedia.org/wiki/Cloud_storage, *[accessed on: 06/05/2014]*

[36] From Wikipedia ,"Hash function", July 2010, http://en.wikipedia.org/wiki/Hash_function *[accessed on: 30-11-2014]*

[37] From Wikipedia, "SHA-2", http://en.wikipedia.org/wiki/SHA-2,

[Accessed on: 30-11-2014]

[38] From Wikipedia, " Secure Hash Algorithm ", http://en.wikipedia.org/wiki/Secure_Hash_Algorithm,

[Accessed on: 30-11-2014]

[39] Xue Jing, Zhang Jian-jun, " A Brief Survey on the Security Model of Cloud Computing", Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science, 2010

[40] How to Install OwnCloud on Windows 7 using Wamp , https://www.youtube.com/watch?v=og8cU2aPmDc,

[Accessed on 08-12-2014]

[41] Zhang lianhong, Chen Hua, " Security Storage in the Cloud Computing: A RSA-based Assumption Data Integrity Check without Original Data ",International Conference on Educational and Information Technology, lCElT, 2010