A NOVEL MECHANISM OF FULLY HOMOMORPHIC ENCRYPTED DATA

K. Mounika [1] Y. SaiTarun[2] V. Harika [3] Ch. SaiRam [4] Ranga swamy Sirisati [5]
[1,2,3,4] B.Tech Scholars, Dept. of CSE, Sri Vasavi institute of engineering and technology, Nandamuru, AP, India
[5]Assistant Professor, Dept. of CSE, Sri Vasavi institute of engineering and technology, Nandamuru, AP, India

ABSTRACT: Cloud storage enables users to remotely store their data and enjoy the on-demand high quality cloud applications without the burden of local hardware and software management. Though the benefits are clear, it inevitably poses new security risks toward the correctness of the data in cloud. To address this problem, the proposed design allows users to audit the cloud storage with very lightweight communication and computation cost by utilizing the new cryptographic technique like homomorphic token and distributed erasure-coded data. By introducing the notion of parallel homomorphic encryption (PHE) schemes, which are encryption schemes that support computation over encrypted data via evaluation algorithms that can be efficiently executed in parallel. In addition, we can hide the function being evaluated with the MapReduce model runnable on Hadoop framework for element testing and keyword search. Analysis shows the proposed scheme is highly efficient and resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks and attempting to strike a balance between security, efficiency and functionality using Kerberos protocol and HDFS system.

KEYWORDS: *Hadoop, Hadoop Distributed File System, Parallel Homomorphic Encryption, Security*.

1. INTRODUCTION

Big data is a term that describes the large volume of data – both structured and unstructured, that inundates a business on a day-to-day basis. But it's not the amount of data that's important. It's what organizations do with the data that matters. Big data can be analyzed for insights that lead to better decisions and strategic business moves. While the term "big data" is relatively new, the act of gathering and storing large amounts of information for eventual analysis is ages old. The concept gained momentum in the early 2000s when industry analysts articulated the now-mainstream definition of big data as the three V's:

Volume – Organizations collect data from a variety of sources, including business transactions, social media and information from sensor or machine-to-machine data. In the past, storing it would've been a problem – but new technologies (such as Hadoop) have eased the burden.

Velocity – Data streams in at an unprecedented speed and must be dealt with in a timely manner. RFID tags, sensors and smart metering are driving the need to deal with torrents of data in near-real time.

Variety – Data comes in all types of formats – from structured, numeric data in traditional databases to unstructured text documents, email, video, audio, stock ticker data and financial transactions.

Hadoop is a complete eco-system of open source projects that provide us the framework to deal with big data. Let's start by brainstorming the possible challenges of dealing with big data (on traditional systems) and then look at the capability of Hadoop solution.

Following are the challenges in dealing with big data :

1. High capital investment in procuring a server with high processing capacity.
2. Enormous time taken
3. In case of long query, imagine an error happens on the last step. You will waste so much time making these iterations.
4. Difficulty in program query building

In the problem of private outsourced computation, a client wishes to delegate the evaluation of a function f on a private input x to an untrusted worker without the latter learning anything about x and f(x). This problem occurs in many applications and, most notably, in the setting of cloud computing, where a provider makes its computational resources available to clients as a service".

One approach to this problem is via the use of homomorphic encryption (HE). An encryption scheme is homomorphic if it supports computation on encrypted data, i.e., in addition to the standard encryption _Work done at Microsoft Research. The decryption algorithms has an evaluation algorithm that takes as input an encryption of some message x and a function f and returns an encryption of f(x). HE schemes can be roughly categorized into two types. The first is arithmetic HE schemes which, in addition to the standard encrypt and decrypt operations, have an add or multiply operation that take as inputs encryptions of messages x1 and x2 and returns encryptions of x1+x2. If an arithmetic HE scheme supports both addition and multiplication, then it can evaluate any arithmetic circuit over encrypted data and we say that it is a fully homomorphic encryption (FHE) scheme. We refer to the second type of HE schemes as non-arithmetic since they do not provide (at least explicitly) addition or multiplication operations.

2. LITERATURE REVIEW

In the existing system the cloud computing, the virtual infrastructure has no parallel computing environment. At such scales, computation is beyond the capabilities of any single machine so it is performed on clusters of machines, i.e., large-scale distributed systems often composed of low-cost unreliable commodity hardware. The problems such as data integrity, data corruption may occur. This leads to unreliability of the cloud computing technology.

In the proposed system, homomorphic encryption with Map Reduce algorithm is used when the computation is performed over massive datasets which will not be possible in the existing system.

The Kerberos protocol and HDFS (Hadoop Distributed File System) are used to enhance high security. HDFS is made up of geographically distributed Data Nodes. Access to these Data Nodes is coordinated by a service called the Name Node. Data Nodes communicate over the network in order to rebalance data and ensure data is replicated throughout the cluster. For my purposes, I will view such a cluster as a system composed of workers and one controller. Given some input, the controller generates n jobs to the workers. Each worker executes its job in parallel and returns some value to the controller who then decides whether to continue the computation or halt.

In this work, we consider the problem of privately outsourcing computation to a cluster of machines.To address this, I introduce parallel homomorphic encryption (PHE) schemes, which are encryption schemes that support computation over encrypted data through the use of an evaluation algorithm that can be efficiently executed in parallel. Using a PHE scheme, a client can outsource the evaluation of a function f on some private input x to a cluster of w machines as follows. The client encrypts x and sends the cipher text and f to the controller. Using the cipher text, the controller generates n jobs that it distributes to the workers and, as above, the workers execute their jobs in parallel. When the entire computation is finished, the client receives a cipher text which it decrypts to recover f(x).

Although the cloud infrastructures are much more powerful and reliable than personal computing devices, broad range of both internal and external threats for data integrity still exist. Examples of outages and data loss incidents of noteworthy cloud storage services appear from time to time. On the other hand, since users may not retain a local copy of outsourced data, there exist various incentives for CSP (Cloud Service Provider) to behave unfaithfully toward the cloud users regarding the status of their outsourced data. For example, to increase the profit margin by reducing cost, it is possible for CSP to discard rarely accessed data without being detected in a timely fashion. Similarly, CSP may even attempt to hide data loss incidents so as to maintain a reputation.

In this paper, we propose an effective and flexible distributed storage verification scheme with explicit dynamic data support to ensure the correctness and availability of users' data in the cloud.

3. PARALLEL HOMOMORPHIC ENCRYPTION (PHE)

3.1. Hadoop framework to cloud-based cluster-computing:

Due to its simplicity and generality, the Map Reduce model has quickly become the standard for working with massive datasets. In fact, it is arguably the most successful and widely used model of parallel computation. In 2008 it was reported that Google processed over 20 petaBytes of data a day using Map Reduce and that Yahoo! deployed a 10,000 core Map Reduce cluster. A Map Reduce algorithm is run by an execution framework that handles the details of distributing work among the machines in the cluster, balancing the workload so as to optimize performance and recovering from failures. The most popular framework is Hadoop which is open source and used by hundreds of large organizations including Amazon, Ebay , Facebook , Yahoo!, Twitter and IBM. Building and maintaining large-scale clusters requires a considerable amount of effort and resources, so a recent trend in cluster-computing has been to make use of cloud infrastructures. Examples include Amazon's Elastic Map Reduce , Cloudera's Hadoop distribution(which can run over several cloud infrastructures) and the recently announced Microsoft Azure Hadoop service. With such services, a client can run a Map Reduce algorithm on massive datasets in the cloud". While these services allow clients to take advantage of all the benefits of cloud computing, they require the client to trust the provider with its data.

3.2. Current level of security in Hadoop:

Current version of hadoop has very basic rudimentary implementation of security which is advisory access control mechanical. Hadoop doesn't strongly authenticate the client, it simply asks the underlying Unix system by executing `whoami` command Any one can communicate directly with a Data node (without the need for communicating with the Name node) and ask for blocks if you have the block location details. Hadoop cluster may be prone to following attacks unauthorized client scan impersonate authorized users and access the cluster. One can get the blocks directly from the Datanodes by bypassing the Namenode. Eaves dropping / sniffing of data packets being sent by Datanodes to client.

3.3. Homomorphic encryption:

As mentioned in section 1, the problem of private outsourced computation can be addressed non-interactively using Homomorphic Encryption (HE).Of course, several semantically secure arithmetic HE schemes are known , including Gentry's breakthrough FHE scheme and its variants . Several non-arithmetic HE schemes are also known including.

3.3.1. Map Reduce:

Map Reduce was introduced by Dean and Ghemawat. In this work, they describe the design and implementation of the Map Reduce framework for processing massive datasets on large-scale systems of loosely coupled machines.

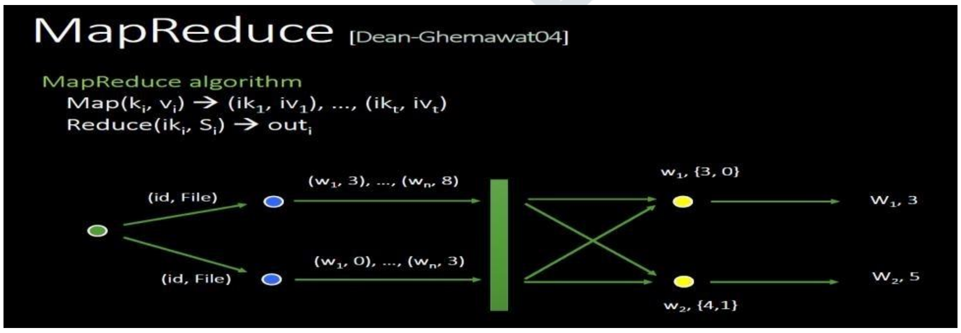

MapReduce [Dean-Ghemawat04]
MapReduce algorithm
Map(k_i, v_i) → (ik_1, iv_1), ..., (ik_t, iv_t)
Reduce(ik_i, S_i) → out_i
(id, File)
(id, File)
(w_1, 3), ..., (w_n, 8)
(w_1, 0), ..., (w_n, 3)
w_1, {3, 0}
w_2, {4,1}
W_1, 3
W_2, 5

Fig. 3.1 Process of Map Reduce

3.3.2. Map Reduce Framework:

Map Reduce is a popular data processing paradigm for efficient and fault tolerant workload distribution in large clusters. A Map Reduce computation has two phases, namely, the Map phase and the Reduce phase. The Map phase splits an input data into a large number of fragments, which are evenly distributed to Map tasks across a cluster of nodes to process. Each Map task takes in a key-value pair and then generates a set of intermediate key-value pairs. After the Map Reduce runtime system groups and sorts all the intermediate values associated with the same intermediate key, the runtime system delivers the intermediate values to Reduce tasks. Each Reduce task takes in all intermediate pairs associated with a particular key and emits a final set of key-value pairs. Map Reduce applies the main idea of moving computation towards data, scheduling map tasks to the closest nodes where the input data is stored in order to maximize data locality. Hadoop is one of the most popular Map Reduce implementations. Both input and output pairs of a Map Reduce application are managed by an underlying Hadoop distributed file system (HDFS). At the heart of HDFS is a single NameNode a master server managing the file system namespace and regulates file accesses. The Hadoop runtime system establishes two processes called Job Tracker and Task Tracker. Job Tracker is responsible for assigning and scheduling tasks; each ask Tracker handles mappers or reducers assigned by Job Tracker. When Hadoop exhibits an overwhelming development momentum, a new Map Reduce programming model Spark attracts researchers' attention . The main abstraction in Spark is a Resilient Distributed Dataset (RDD), which offers good fault tolerance and allows jobs to perform computations in memory on large clusters. Thus, Spark becomes an attractive programming model to iterative Map Reduce algorithms. We decide to develop FiDoop-DP on Hadoop clusters; in a future study, we plan to extend FiDoop-DP to Spark to gain further performance improvement.

Perhaps the simplest Map Reduce algorithm is to determine frequency counts, i.e., the number times a keyword occurs in a document collection. The parse algorithm takes the document collection (D1,…,Dn) as input and outputs a set of input pairs $(I,D_i)_i$. Each mapper receives an input pair (i;Di) and outputs a set of intermediate pairs (word count for each document) wj found in Di. All the intermediate pairs are then partitioned by the partition operation into sets.

3.3.3. Parallel FP-Growth Algorithm:

In this existing study, we focus on a FP-Growth algorithm called Parallel FP. FP-Growth efficiently discovers frequent item sets by constructing and mining a compressed data structure (i.e., FP-tree) rather than an entire database. PFP was designed to address the synchronization issues by partitioning transaction database into independent partitions, because it is guaranteed that each partition contains all the data relevant to the features (or items) of that group. Given a transaction database DB, The following figure depicts the process flow of Parallel FP Growth implemented in Mahout. The parallel algorithm consists of four steps, three of which are Map Reduce jobs.

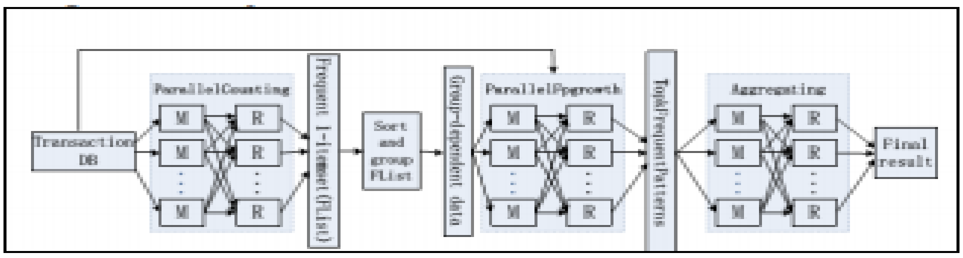


Fig.3.2. Process flow of Parallel FP Growth

Step 1. Parallel Counting: The first Map Reduce job counts the support values of all items residing in the database to discover all frequent items or frequent 1-itemsets in parallel. It is worth noting that this step simply scans the database once.

Step 2. Sorting frequent 1-itemsets to F List: The second step sorts these frequent 1-itemsets in a decreasing order of frequency; the sorted frequent 1-itemsets are cached in a list named F List. Step 2 is a non-Map Reduce process due to its simplicity as well as the centralized control.

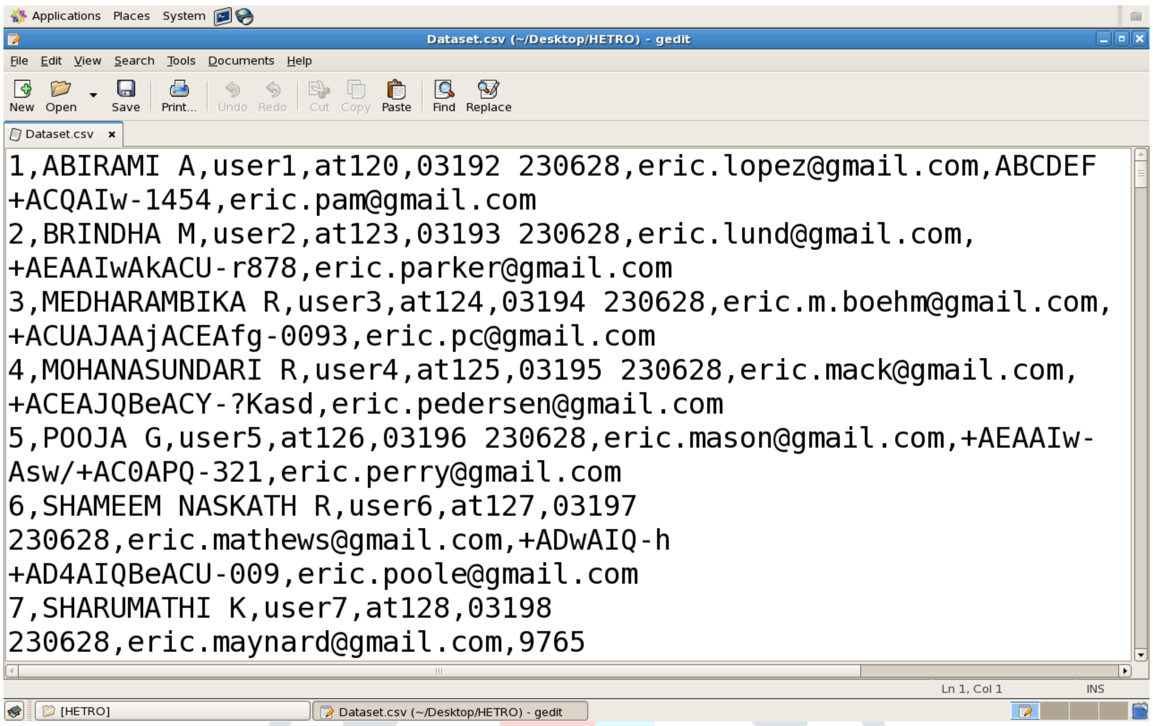
Step 3. Parallel FP-Growth: This is a core step of PFP, where the map stage and reduce stage perform the following two important functions. Mapper - Grouping items and generating group-dependent transactions and Reducer - FP-Growth on group dependent partitions. Local FP Growth is conducted to generate local frequent item sets. Each reducer conducts local FP Growth by processing one or more group-dependent partition one by one, and discovered patterns are output in the final.

Step 4. Aggregating: The last Map Reduce job produces final results by aggregating the output generated in Step

Security Implementation

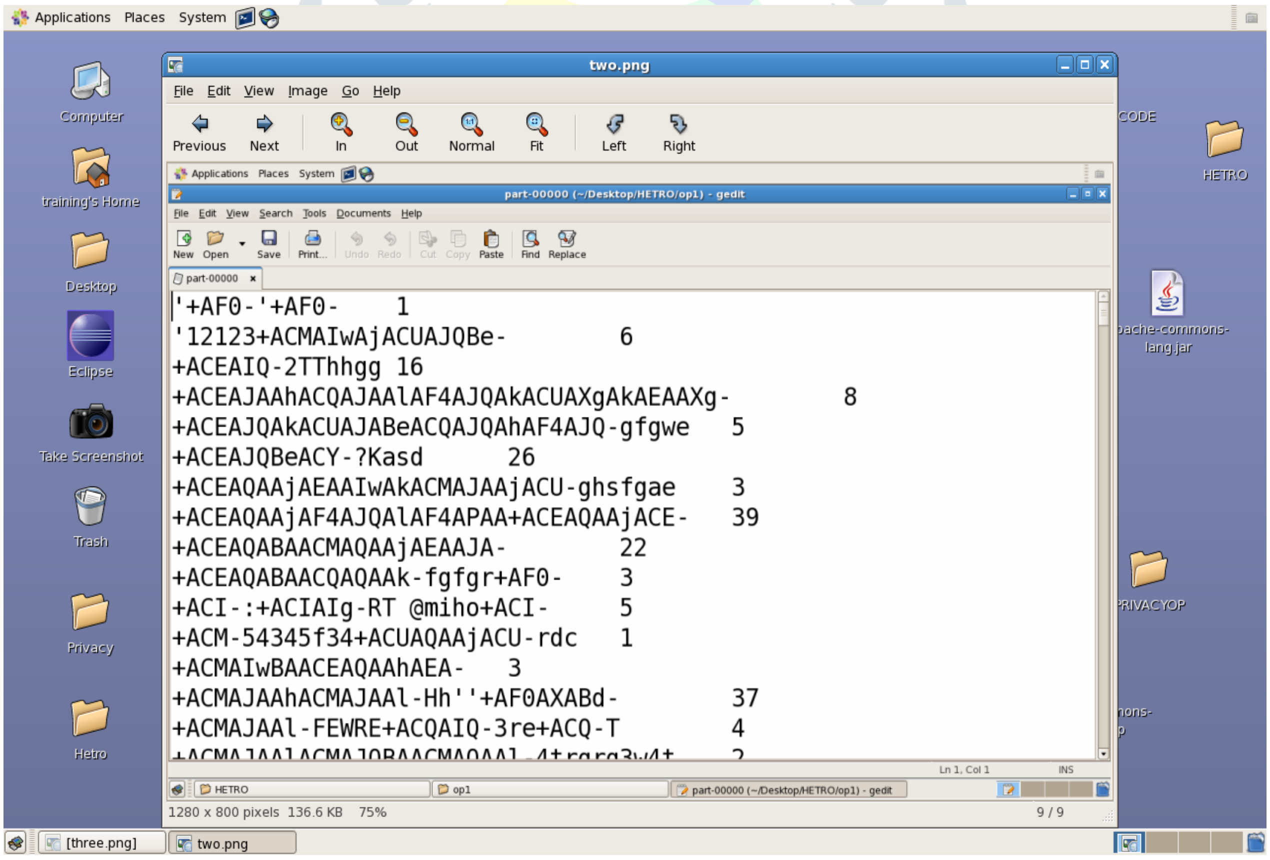
A network authentication protocol is used as private-key cryptography for providing authentication across open the cloud computing environment. It mediates authentication through a trusted third party. It provides authentication for the confirmation that a user who is requesting services is a valid user of the network services requested. The next is Authorization, the granting of specific types of service to a user, based on their authentication, what services they are requesting, and the current system state. At last accounting, the tracking of the consumption of network resources by users.

4. RESULTS


Applications Places System
Dataset.csv (~/Desktop/HETRO) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
Dataset.csv
Ln 1, Col 1 INS
[HETRO]
Dataset.csv (~/Desktop/HETRO) - gedit

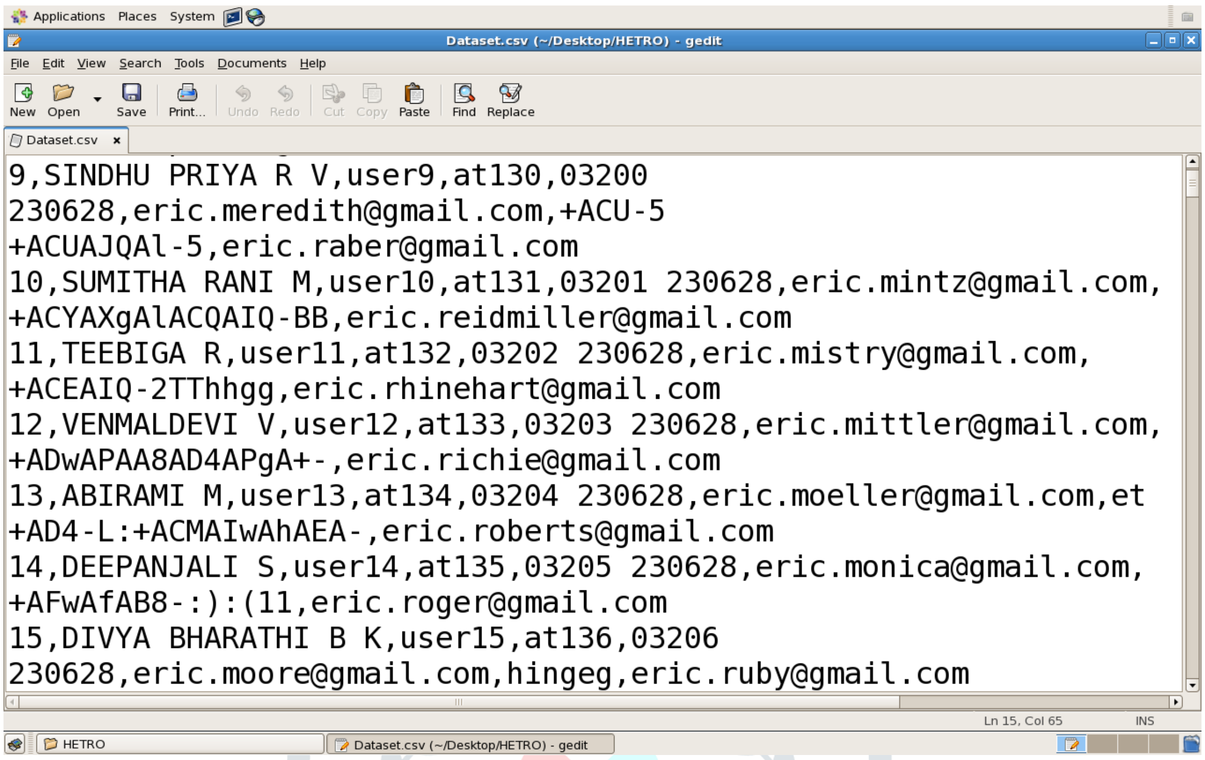
```
1,ABIRAMI A,user1,at120,03192 230628,eric.lopez@gmail.com,ABCDEF
+ACQAIw-1454,eric.pam@gmail.com
2,BRINDHA M,user2,at123,03193 230628,eric.lund@gmail.com,
+AEAAIwAkACU-r878,eric.parker@gmail.com
3,MEDHARAMBIKA R,user3,at124,03194 230628,eric.m.boehm@gmail.com,
+ACUAJAAjACEAfg-0093,eric.pc@gmail.com
4,MOHANASUNDARI R,user4,at125,03195 230628,eric.mack@gmail.com,
+ACEAJQBeACY-?Kasd,eric.pedersen@gmail.com
5,POOJA G,user5,at126,03196 230628,eric.mason@gmail.com,+AEAAIw-
Asw/+AC0APQ-321,eric.perry@gmail.com
6,SHAMEEM NASKATH R,user6,at127,03197
230628,eric.mathews@gmail.com,+ADwAIQ-h
+AD4AIQBeACU-009,eric.poole@gmail.com
7,SHARUMATHI K,user7,at128,03198
230628,eric.maynard@gmail.com,9765
```

Fig.4.1.Input Data


Applications Places System
two.png
File Edit View Image Go Help
Previous Next In Out Normal Fit Left Right
Computer
training's Home
Desktop
Eclipse
Take Screenshot
Trash
Privacy
Hetro
CODE
HETRO
pache-commons-lang.jar
PRIVACYOP
part-00000 (~/Desktop/HETRO/op1) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
part-00000
Ln 1, Col 1 INS
HETRO
op1
1280 x 800 pixels 136.6 KB 75%
9 / 9
[three.png]
two.png

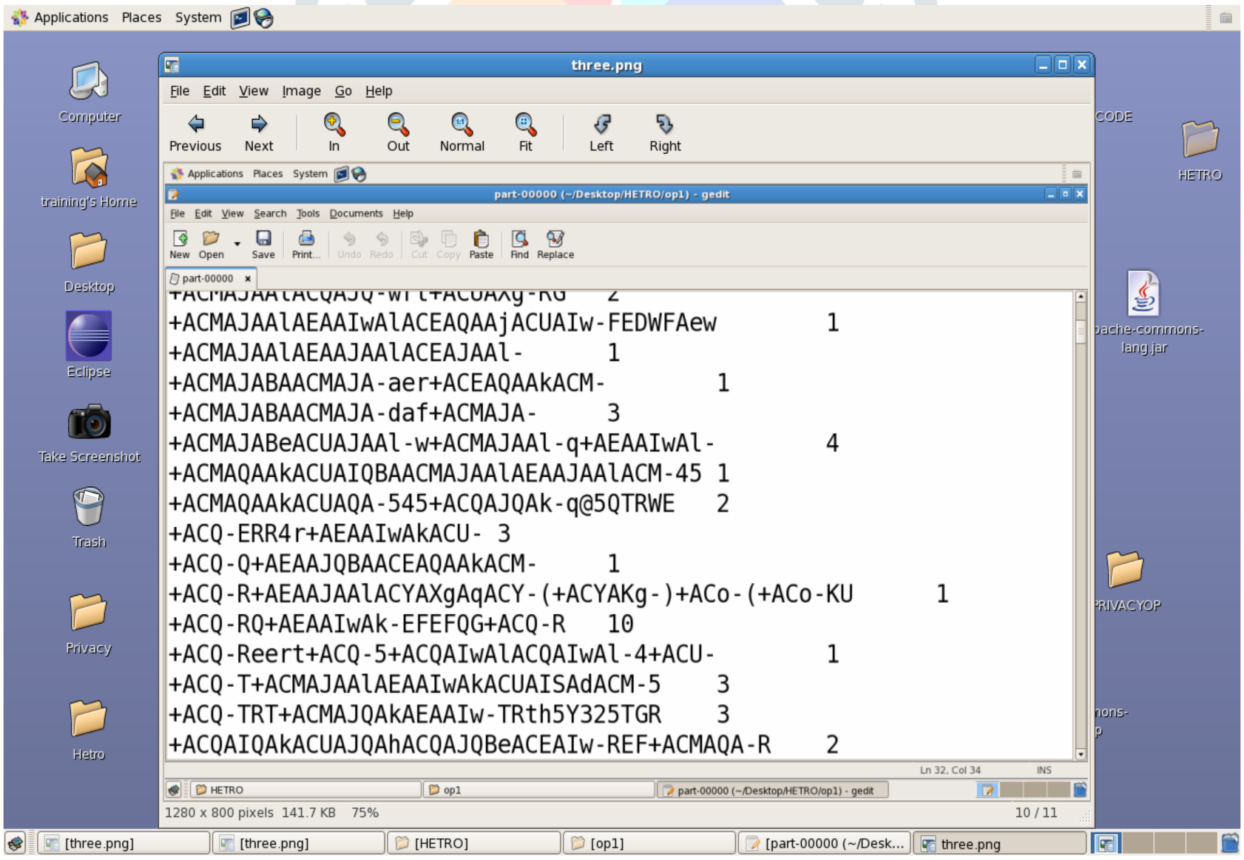
```
'+AF0-'+AF0-     1
'12123+ACMAIwAjACUAJQBe-        6
+ACEAIQ-2TThhgg 16
+ACEAJAAhACQAJAAlAF4AJQAkACUAXgAkAEAAXg-        8
+ACEAJQAkACUAJABeACQAJQAhAF4AJQ-gfgwe   5
+ACEAJQBeACY-?Kasd      26
+ACEAQAAjAEAAIwAkACMAJAAjACU-ghsfgae    3
+ACEAQAAjAF4AJQAlAF4APAA+ACEAQAAjACE-   39
+ACEAQABAACMAQAAjAEAAJA-        22
+ACEAQABAACQAQAAk-fgfgr+AF0-    3
+ACI-:+ACIAIg-RT @miho+ACI-     5
+ACM-54345f34+ACUAQAAjACU-rdc   1
+ACMAIwBAACEAQAAhAEA-   3
+ACMAJAAhACMAJAAl-Hh''+AF0AXABd-        37
+ACMAJAAl-FEWRE+ACQAIQ-3re+ACQ-T        4
+ACMAJAAlACMAJQBAACMAQAAl-4trgrg3v4t    2
```

Fig.4.2. Fully Homomorphic Encrypted Data



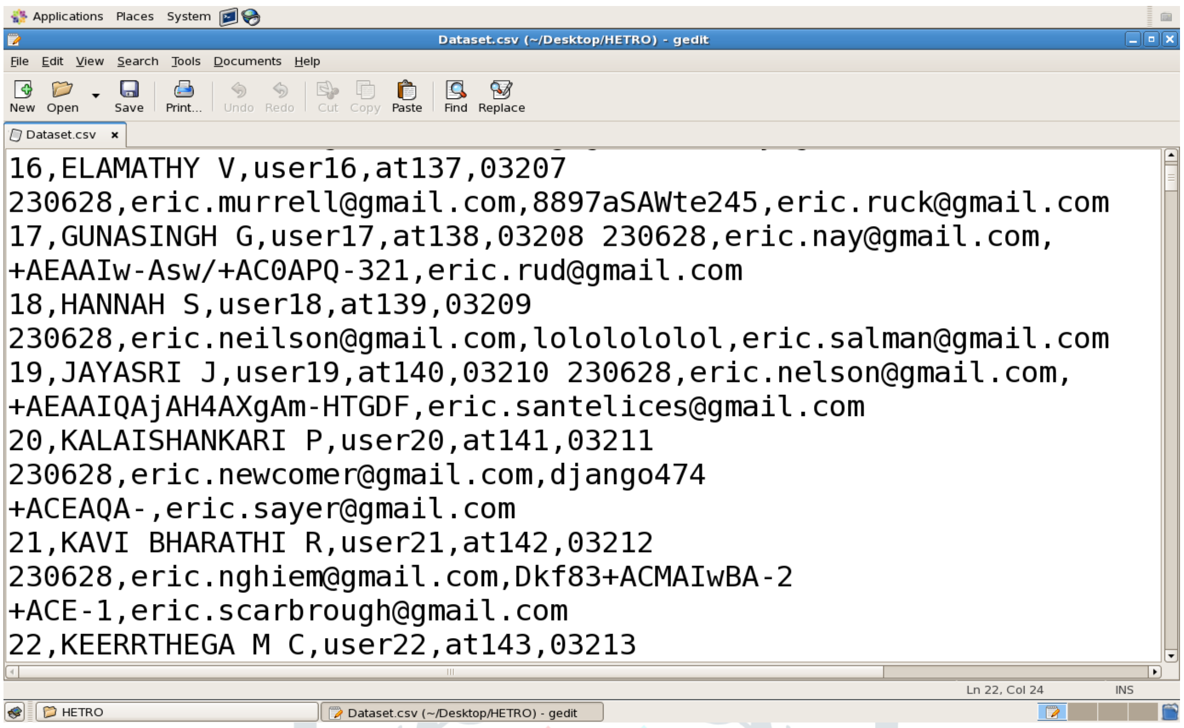
```
9,SINDHU PRIYA R V,user9,at130,03200
230628,eric.meredith@gmail.com,+ACU-5
+ACUAJQAl-5,eric.raber@gmail.com
10,SUMITHA RANI M,user10,at131,03201 230628,eric.mintz@gmail.com,
+ACYAXgAlACQAIQ-BB,eric.reidmiller@gmail.com
11,TEEBIGA R,user11,at132,03202 230628,eric.mistry@gmail.com,
+ACEAIQ-2TThhgg,eric.rhinehart@gmail.com
12,VENMALDEVI V,user12,at133,03203 230628,eric.mittler@gmail.com,
+ADwAPAA8AD4APgA+-,eric.richie@gmail.com
13,ABIRAMI M,user13,at134,03204 230628,eric.moeller@gmail.com,et
+AD4-L:+ACMAIwAhAEA-,eric.roberts@gmail.com
14,DEEPANJALI S,user14,at135,03205 230628,eric.monica@gmail.com,
+AFwAfAB8-:):(11,eric.roger@gmail.com
15,DIVYA BHARATHI B K,user15,at136,03206
230628,eric.moore@gmail.com,hingeg,eric.ruby@gmail.com
```

Fig.4.3. Input Data



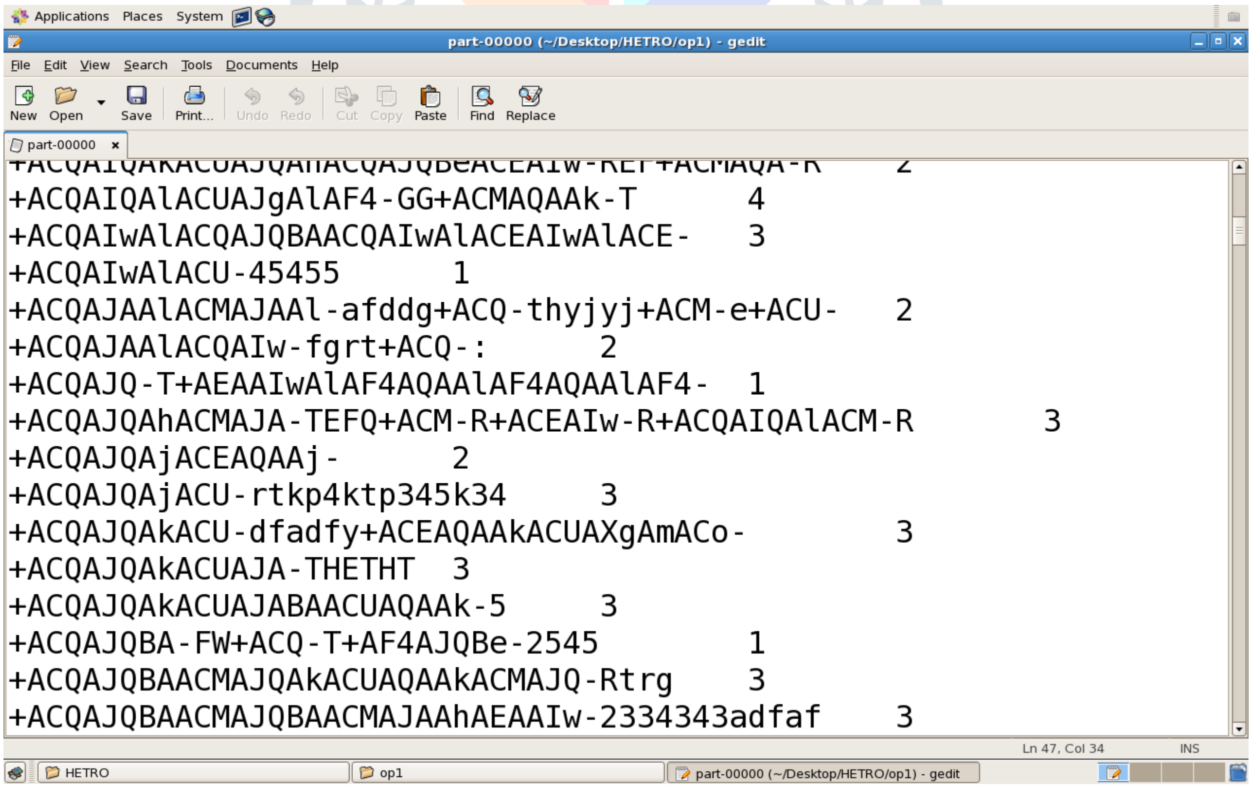
```
+ACMAJAAlACQAJQ-wIt+ACUAXg-RG   2
+ACMAJAAlAEAAIwAlACEAQAAjACUAIw-FEDWFAew        1
+ACMAJAAlAEAAJAAlACEAJAAl-      1
+ACMAJABAACMAJA-aer+ACEAQAAkACM-        1
+ACMAJABAACMAJA-daf+ACMAJA-     3
+ACMAJABeACUAJAAl-w+ACMAJAAl-q+AEAAIwAl-        4
+ACMAQAAkACUAIQBAACMAJAAlAEAAJAAlACM-45 1
+ACMAQAAkACUAQA-545+ACQAJQAk-q@5QTRWE   2
+ACQ-ERR4r+AEAAIwAkACU- 3
+ACQ-Q+AEAAJQBAACEAQAAkACM-     1
+ACQ-R+AEAAJAAlACYAXgAqACY-(+ACYAKg-)+ACo-(+ACo-KU       1
+ACQ-RQ+AEAAIwAk-EFEFQG+ACQ-R   10
+ACQ-Reert+ACQ-5+ACQAIwAlACQAIwAl-4+ACU-        1
+ACQ-T+ACMAJAAlAEAAIwAkACUAISAdACM-5    3
+ACQ-TRT+ACMAJQAkAEAAIw-TRth5Y325TGR    3
+ACQAIQAkACUAJQAhACQAJQBeACEAIw-REF+ACMAQA-R    2
```

Fig.4.4. Fully Homomorphic Encrypted Data


Applications Places System
Dataset.csv (~/Desktop/HETRO) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
Dataset.csv
Ln 22, Col 24 INS
HETRO
Dataset.csv (~/Desktop/HETRO) - gedit

```
16,ELAMATHY V,user16,at137,03207
230628,eric.murrell@gmail.com,8897aSAWte245,eric.ruck@gmail.com
17,GUNASINGH G,user17,at138,03208 230628,eric.nay@gmail.com,
+AEAAIw-Asw/+AC0APQ-321,eric.rud@gmail.com
18,HANNAH S,user18,at139,03209
230628,eric.neilson@gmail.com,lolololol,eric.salman@gmail.com
19,JAYASRI J,user19,at140,03210 230628,eric.nelson@gmail.com,
+AEAAIQAjAH4AXgAm-HTGDF,eric.santelices@gmail.com
20,KALAISHANKARI P,user20,at141,03211
230628,eric.newcomer@gmail.com,django474
+ACEAQA-,eric.sayer@gmail.com
21,KAVI BHARATHI R,user21,at142,03212
230628,eric.nghiem@gmail.com,Dkf83+ACMAIwBA-2
+ACE-1,eric.scarbrough@gmail.com
22,KEERRTHEGA M C,user22,at143,03213
```

Fig.4.5. Input Data


Applications Places System
part-00000 (~/Desktop/HETRO/op1) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
part-00000
Ln 47, Col 34 INS
HETRO
op1
part-00000 (~/Desktop/HETRO/op1) - gedit

```
+ACQAIQAlACUAJgAlAF4-GG+ACMAQAAk-T        4
+ACQAIwAlACQAJQBAACQAIwAlACEAIwAlACE-    3
+ACQAIwAlACU-45455       1
+ACQAJAAlACMAJAAl-afddg+ACQ-thyjyj+ACM-e+ACU-    2
+ACQAJAAlACQAIw-fgrt+ACQ-:       2
+ACQAJQ-T+AEAAIwAlAF4AQAAlAF4AQAAlAF4-   1
+ACQAJQAhACMAJA-TEFQ+ACM-R+ACEAIw-R+ACQAIQAlACM-R        3
+ACQAJQAjACEAQAAj-       2
+ACQAJQAjACU-rtkp4ktp345k34      3
+ACQAJQAkACU-dfadfy+ACEAQAAkACUAXgAmACo-        3
+ACQAJQAkACUAJA-THETHT   3
+ACQAJQAkACUAJABAACUAQAAk-5      3
+ACQAJQBA-FW+ACQ-T+AF4AJQBe-2545        1
+ACQAJQBAACMAJQAkACUAQAAkACMAJQ-Rtrg     3
+ACQAJQBAACMAJQBAACMAJAAhAEAAIw-2334343adfaf     3
```

Fig.4.6. Fully Homomorphic Encrypted Data

5. CONCLUSION

By utilizing the homomorphic token with distributed verification of erasure coded data, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., whenever data corruption has been detected during the storage correctness verification across the distributed servers, we can almost guarantee the simultaneous identification of the misbehaving server(s). Through detailed security and extensive experiment results, we show that our scheme is highly efficient and resilient to Byzantine failure, malicious data modification attack, and even server colluding attacks.

REFERENCES

[1] Powered by hadoop. See http://wiki.apache.org/hadoop/PoweredBy.
[2] Yahoo! launches world's largest hadoop production application. See http://developer.yahoo.net/blogs/hadoop/2008/02/yahooworlds-largest-production-hadoop.html, 2008.
[3] M. Abadi, J. Feingenbaum, and J. Killian. On hiding information from an oracle. In ACM Symposium on Theory of Computing (STOC 1987), pages 195{203, 1987.
[4] B. Applebaum, Y. Ishai, and E. Kushilevitz. Computationally private randomizing polynomials and their applications. Journal of Computational Complexity, 15(2), 2006.
[5] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC0. SIAM Journal of Computation, 36(4):845{888, December 2006.
[6] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography with constant input locality. In Advances in Cryptology - CRYPTO '07, Lecture Notes in Computer Science, pages 92{110. Springer-Verlag, 2007. 19 (CCS '07), pp. 584-597, Oct. 2007. [9]B. Krebs, "Payment Processor Breach May Be Largest Ever," http://voices.washingtonpost.com/securityfix/2009/ 01/ payment_processor_breach_may_b.html, Jan. 2009.
[7] A. Juels and B.S. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 584-597, Oct. 2007.
[8] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 598-609, Oct. 2007.
[9] M.A. Shah, M. Baker, J.C. Mogul, and R. Swaminathan, "Auditing to Keep Online Storage Services Honest," Proc.
[10] 11th USENIX Workshop Hot Topics in Operating Systems (HotOS '07), pp. 1-6, 2007.
[11] M.A. Shah, R. Swaminathan, and M. Baker, "Privacy-PreservingAudit and Extraction of Digital Contents," Cryptology ePrintArchive, Report 2008/186, http://eprint.iacr.org, 2008.
[12] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. Fourth Int'l Conf.Security and Privacy in Comm. Netowrks (SecureComm '08), pp. 1-10,2008.
[13] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing," Proc. 14th European Conf. Research in Computer Security (ESORICS '09), pp. 355-370, 2009.
[14] Y. Dodis, S. Vadhan, and D. Wichs, "Proofs of Retrievability via Hardness Amplification," Proc. Sixth Theory of Cryptography Conf.(TCC '09), Mar. 2009.
[15] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," IEEE Trans. Parallel and Distributed Systems, vol. 22,no. 5, pp. 847-859, 2011.
[16] C. Wang, S.S.M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy- Preserving Public Auditing for Secure Cloud Storage," IEEE Trans. Computers, preprint, 2012, doi:10.1109/TC.2011.245.
[17] K.D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," Proc. ACM Conf.
Computer and Comm. Security (CCS '09), pp. 187-198, 2009.
[18] T. Schwarz and E.L. Miller, "Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage," Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS '06), pp. 12-12, 2006.
[19] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard, "A Cooperative Internet Backup Scheme," Proc. USENIX Ann.
Technical Conf. (General Track), pp. 29-41, 2003.
[20] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance and Proactive Recovery," ACM Trans. Computer Systems, vol. 20, no. 4, pp. 398-461, 2002.