Global Journal of Computer Science and Technology
Cloud and Distributed
Volume 13 Issue 4 Version 1.0 Year 2013
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

# Safeguarding the Liabilities of Data Accessing in Cloud Computing

By Kethan Harish Yekula, Dr. Y. Venkateshwarlu & Suneel Kumar Badugu
*Jawaharlala Nehru Technological University Kakinada (JNTUK), India*

*Abstract* - Cloud computing is the process of providing the virtualized services over the internet. The space in the web commonly known as Cloud has been monitored by service provider. In a real time scenario, a user registers for a particular service and shares his data as well as access credential policies with CSP (cloud service provider). Though cloud computing has got major flexibility in data accessing, users are very much concerned about their data security as it may be mislead by service providers. They may share the owner's data to unauthenticated persons. This is a big threat to the data owners. In this paper a modern approach, is proposed namely Cloud Information Accountability (CIA) framework, and based on the notion of data liability. We identify the common requirements and develop several guidelines to achieve data accountability in the cloud. Once the data owner provides data, the service provider will have full access and permission rights, on the data. Using traditional access control mechanisms, after data rights are permitted, the data is in the hands of the service provider. We propose an algorithm, which gives the details of people accessing the data using the automated logging details through the JAR files.

*Keywords : cloud computing, logging, privacy, security, data sharing, information accountability framework.*

*GJCST-B Classification : C.1.4*

SAFEGUARDING THE LIABILITIES OF DATA ACCESSING IN CLOUD COMPUTING

# Safeguarding the Liabilities of Data Accessing in Cloud Computing

Kethan Harish Yekula [α], Dr. Y. Venkateshwarlu [σ] & Suneel Kumar Badugu [ρ]

*Abstract -* Cloud computing is the process of providing the virtualized services over the internet. The space in the web commonly known as Cloud has been monitored by service provider. In a real time scenario, a user registers for a particular service and shares his data as well as access credential policies with CSP (cloud service provider). Though cloud computing has got major flexibility in data accessing, users are very much concerned about their data security as it may be mislead by service providers. They may share the owner's data to unauthenticated persons. This is a big threat to the data owners. In this paper a modern approach, is proposed namely Cloud Information Accountability (CIA) framework, and based on the notion of data liability. We identify the common requirements and develop several guidelines to achieve data accountability in the cloud. Once the data owner provides data, the service provider will have full access and permission rights, on the data. Using traditional access control mechanisms, after data rights are permitted, the data is in the hands of the service provider. We propose an algorithm, which gives the details of people accessing the data using the automated logging details through the JAR files.

*Keywords : cloud computing, logging, privacy, security, data sharing, information accountability framework.*

## I. Introduction

The *Cloud Information Accountability framework (CIA)* proposed in this work conducts automated logging and distributed auditing of relevant access performed by any object, carried out at any point of time at any CSP's. The CIA concept has two major components: logger and log harmonizer. The JAR file holds a set of simple access control rules specifying whether and how the cloud servers and possibly other data stakeholders are authorized to access the content. Once the login credentials are matched, the service provider accesses the data hidden in the JAR. Based on the settings mentioned during creation, the Java Archive file gives usage control along with log options, for every log, the data is accessed and the JAR generates a record. Cloud computing presents a replacement thanks to resources. The consumption model for IT services on the net, by providing support for ad-hoc increasable randomly virtualizing resources are used as a service over the internet. Knowledge handling is outsourced by the direct cloud service supplier (CSP) to different entities within the cloud and theses entities forward the applications to others, and so on. And, permissions are provided to enter and leave the cloud storage. As a result, knowledge handling within the cloud goes through a posh and dynamic hierarchal service chain that does not exist in standard environments.

To overcome the above mentioned issues, we tend to propose a unique approach, specifically Cloud information Accountability (CIA) framework, supporting the notation of knowledge responsibility. Some of the common needs are determined and various tips are suggested to attain knowledge answerability within the cloud. For example A user, United Nations agency signed to an explicit cloud service, typically has to send his/her knowledge moreover as associated access management policies to the service supplier. Once the info is received, the CSP can get access permission rights, like scan, write, and copy, on the info exploitation standard access management mechanisms, once the access rights are granted, the info are going to be totally offered at the service supplier.

The CIA framework projected during this work conducts machine-controlled work and distributed auditing of relevant access performed by any entity, applied at any purpose to any cloud service applies its two major components: logger and log harmonizer.

The JAR file includes a group of straightforward access management rules specifying whether or not and the way the cloud servers and presumably different knowledge stakeholders are approved to access the data. After the verification is done, the service suppliers are given the permission to access the information closed within the JAR looking on the configuration settings outlined at the time of creation, the JAR can offer usage management related to work. Whenever the associates access the information from cloud, The JAR automatically generates a log report.

## II. Literature Survey

### a) *Provenance Management in Curated Databases (Peter Buneman, Adriane P. Chapman, James Cheney)*

Curated databases in bioinformatics and other disciplines are the result of a great deal of manual annotation, correction and transfer of data from other sources. Provenance information concerning the creation, attribution, or version history of such data is crucial for assessing its integrity and scientific value. General-purpose database systems provide little

*Author : Jawaharlala Nehru Technological University Kakinada (JNTUK). E-mail : kethan.harish@gmail.com*

support for tracking provenance, especially when data moves among databases. This paper investigates general-purpose techniques for recording provenance for data that is copied among databases. We describe an approach in which we track the user's actions while browsing source databases and copying data into a curate database, in order to record the user's actions in a convenient, query able form. We present an implementation of this technique and use it to evaluate the feasibility of database support for provenance management. Our experiments show that although the overhead of a native approach is high, it can be decreased to an acceptable level using simple optimizations.

*b) The Advantages of Elliptic Curve Cryptography for Wireless Security Kristin Lauter, Microsoft Corporation*

This article provides an overview of elliptic curves and their use in cryptography. The focus is on the performance advantages to be obtained in the wireless environment by using elliptic curve cryptography instead of a traditional cryptosystem like RSA. Specific applications to secure messaging and identity-based encryption are discussed.

*c) Identity-Based Encryption from the Weil Pairing (Dan Boneh, And Matt Franklin)*

We propose a fully functional identity-based encryption scheme (IBE). The scheme has chosen cipher text security in the random oracle model assuming an elliptic curve variant of the computational Diffe-Hellman problem. We give precise definitions for secure identity based encryption schemes and give several applications for such systems.

*d) Verifiable Security of Boneh-Franklinidentity-Based Encryption (Gilles Barthe, FedericoOlmedo, and Santiago ZanellaBeguelin)*

Identity-based encryption (IBE) allows one party to send ciphered messages to another using an arbitrary identity string as an encryption key. Since IBE does not require prior generation and distribution of keys, it greatly simplifies key management in public-key cryptography. Although Shamir introduced the concept of IBE in 1981, constructing a practical IBE scheme remained an open problem for years. The first satisfactory solution was proposed by Boneh and Franklin in 2001 and constitutes one of the most prominent applications of pairing- based cryptography. We present a game-based machine-checked reduction of the security of the Boneh-Franklin IBE scheme to the Bilinear Diffie-Hellman assumption, and analyze its tightness by providing an exact security bound. Our proof simplifies and clarifies the original proof by Boneh and Franklin and automatically verifies by running a trusted checker.

## III. Existing System

- Data handling in the cloud undergoes a very complex and dynamic structural service chain.
- In conventional environments these kind of services does not go well in practice.
- For data auditing ordinary web framework is implemented.
- To get request and responses normal web services are used.

*a) Disadvantages*

- User's data is not secured. Authentication or security not provided.
- The expensive resources are needed for implementation
- Not suitable for small and medium level storage users.

*b) Proposed System*

- We propose a modern approach, namely Cloud Information Accountability (CIA) framework, based on the notion of data liability.
- The suggested CIA framework provides end-to end accountability in a highly distributed fashion.
- A detailed security analysis is provided, strengths and reliabilities are discussed and our robust architecture in the face of various nontrivial attacks implements Java Running Environment.
- Apart from creating a class file which authenticates the servers or the users, another class file generates the correct inner JAR, a third class file which checks the JVM's validity using oblivious hashing.
- To limit the access for security purpose, timer mechanism is proposed
- Securing the JVM for making software tamper resistance capabilities to JAR file. It provides integrity, confidentiality to JAR.

*c) Advantages*

- The novel features of the CIA framework lies in its ability to maintain lightweight and powerful accountability that combines aspects of access control, usage control and authentication.
- This technique defends against man in the middle attack, dictionary attack, Disassembling Attack, Compromised JVM Attacks.
- It is suitable for limited and large number of storages.

Architecture Diagram

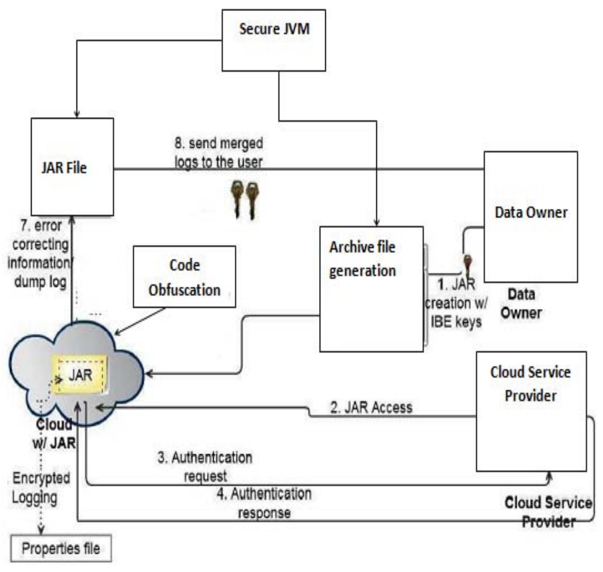

*Figure 1 :* Architecture Diagram

## IV. Related Work

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.

### *a) JAR Generation*

The JAR file contains a set of access control policies mentioning whether and how the cloud servers and possibly other data interested party (users, companies) are authorized to access the information. Depending on the configuration settings, the JAR will provide usage control combined with login credentials.

### *b) Logger Creation*

To conduct automated logging, the programming capabilities of the JAR files are leveraged. User's info and related data items are stored in nested Java JAR file, which is a logger component. The outer JAR provides authentication to entities for accessing the data stored in the JAR file. In this situation, the data owners do not know the exact CSPs who handle data. Hence, authentication is mentioned based on the server's functionality. The data owner can give permissions in user-centric terms as opposed to the usual code-centric security offered by Java, using its Authentication and Authorization Services. Moreover, the outer JAR is also heads the selection of correct inner JAR according to the identity of the entities that requests the data.

### *c) Log Record Generation*

Logger component generates the log records. Logging occurs at any access to the data in the JAR, and consecutively adds the entities in the order of creation LR= (r1; . . . ; rki. Each record is encrypted one by one and updated to the log file. During the read-only request, the inner JAR decrypts the data and a temporary decrypted file is created. This file is shown to the entity using the Java application viewer and suppose it is displayed to a user. Presenting the data in the Java application, he un checks the methods that copy by using hot keys such as Print Screen.

### *a) Mode Setting*

The data owners are timely and accurately informed about their data usage, our distributed logging mechanism complements an unique auditing mechanism. We support two complementary auditing modes:

i Push mode
ii Pull mode.

- *Push Mode:* In this mode, the harmonizer pushes the logs to the data owner (or auditor) in a timely manner. The push action is invoked by either type of the following two events: one is that the time elapses for a certain period according to the temporal timer inserted as part of the JAR file; the other is that the JAR file exceeds the size stipulated by the content owner at the time of creation.
- *Pull Mode:* This mode allows auditors to retrieve the logs anytime when they want to check the recent access to their own data.

### *b) Algorithm*

The concept of push-pull strategies is very beneficial when more amounts of data are accessed in short time. At this scenario, if the data is not sent out random enough, the logging file becomes huge, which increases the operation expenses like data copying. The data owners prefer the push mode and want to keep track of the data usage consistently over time. For such data owners, receiving the logs automatically can lighten the load of the data analyzers. The maximum size at which logs are pushed out is a parameter which can be easily configured while creating the logger component. The pull strategy is most needed when the data owner suspects some misuse of his data; the pull mode allows him to monitor the usage of his content immediately. A hybrid strategy can actually be implemented to benefit of the consistent information offered by pushing mode and the convenience of the pull mode.

### *c) Logging Mechanism*

- The Logger Structure
- Log Record Generation
- Dependability of Logs
- JARs Availability
- Log Correctness

### d) *The Logger Structure*

To conduct the automated logging, the programming capabilities of the JAR file are leveraged.. A logger component is a nested Java JAR file holds a user's data items and related log files.

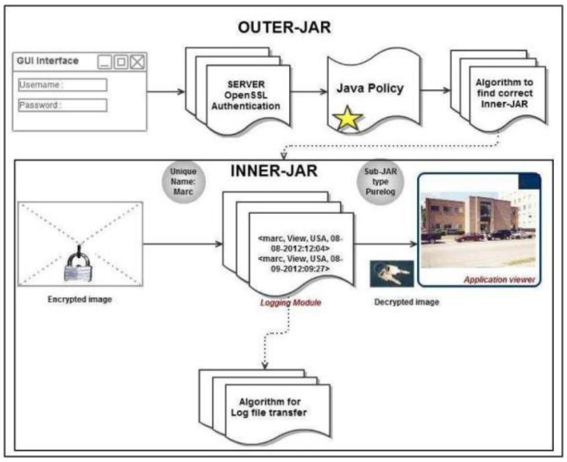
OUTER-JAR
GUI Interface
Username:
Password:
SERVER OpenSSL Authentication
Java Policy
Algorithm to find correct Inner-JAR
INNER-JAR
Unique Name: Marc
Sub-JAR type Purelog
<marc, View, USA, 08-08-2012:12:04>
<marc, View, USA, 08-09-2012:09:27>
Encrypted image
Logging Module
Decrypted image
Application viewer
Algorithm for Log file transfer

*Figure 2 :* Logger Structure

The outer JAR takes care of authentication of entities for accessing the data stored in the JAR file. In this scenario, the data owners may not know the exact CSPs who hold the data. Hence, authentication is specified according to the servers' workability For example, a policy may state that Server X is allowed to download the data if it is a storage server. As discussed below, the outer JAR may gain access control to enforce the data owner's parameters, mentioned as policy of java, which is implemented on the data.

What permissions are available to access a particular code in the Java Environment is taken care by the Java policies. These access rights represent the same File System Permissions. However, the data owner can mention the access permissions in user-centric terms as inverse to the usual code-centric security in Java, using its authorization and authentication Services. Moreover, the outer JAR takes the responsibility of selecting the correct inner JAR according to the identity of the entity who requests the data.

To facilitate retrieval of log files and display enclosed data in a suitable format, and to maintain a log file for each encrypted item, Each inner JAR holds the encrypted data, class files. Here two options are supported:

- *Pure Log :* Its records every access to the data and are used for pure auditing purpose
- *Access Log :* It performs two methods logging actions and enforcing access control. In case if a request is denied, the JAR records the request made time If the access request is granted, the JAR record the access information additionally along with the allowed time period access.

These two logging methods provokes the data owner to implement certain access conditions either proactively (in case of Access Logs) or reactively (in case of Pure Logs). For example, services like billing require use of Pure Logs. Access Logs are needed by services, which compel service-level agreements such as limiting the access to sensitive information. To perform these tasks, the inner JAR writes the log record using the class files and another class file agrees with the log harmonizer, the third class file which is an encrypted file that displays or copies the data from the server downloading the data (based on whether we have a Pure Log, or an Access Log), and the public key of the IBE key pair that is necessary for encrypting the log records.

System never stores the secret keys. The outer JAR may contain one or more inner JARs, apart from that a class file for authenticating the servers or the users, and the another one used to find the correct inner JAR, a third class file using oblivious hashing, checks the JVM's validity. Moreover, class file manages the Graphical User Interface for user authentication and the Java Policy.

### e) *Log Record Generation*

Logger Component generates log records. Any access to the data in the JAR requires logging information, and new log entries are added in a consecutive manner, in order of creation LR= (r1; . . . ; rki). Each record ri is encrypted one by one and added to the log file. To make sure that the log records are correct, Access time, locations as well as actions are verified. In particular, the time of access is determined using the Network Time Protocol (NTP) [15] to avoid suppression of the correct time by a malicious entity. By using the IP address, the Cloud Service Providers location is traced out. The JAR performs IP lookup and finds the most probable location of the CSP using the IP address range. For determining locations, advanced techniques are implemented [10]. Similarly, if a trusted time stamp management infrastructure can be set up or leveraged, it can be used to record the time stamp in the accountability log [2]. The most critical part is to log the actions on the users' data. In the current system, we support four types of actions, i.e., Act has one of the following four values [1]: view, download, timed access, and Location-based access. For each action, we propose a specific method to correctly record or enforce it depending on the type of the logging module, which are elaborated as follows:

- *View*

The entity (e.g., the cloud service provider) has only read permissions and cannot save the raw information permanently. In this type of activity, pure log edits the log record about the access while the Access Logs using the enclosed access control module compels the action. We know that the inner JAR stores and encrypts the data. When the access request is read only, on the fly a temporary decrypted file of data is created by inner JAR. In case the human user accesses the file, by using the Java application viewer the hidden file is shown to the entity viewer. Utilising the Java Applications data, viewer uncheck the copying methods by using some keys such as print screen. And, the data will be hidden to prevent to avoid the usage of some print screen capture software when the application viewing screen focus goes out. When the content is presented to Cloud Service Provider he sees it on the command line using the headless mode in Java

*Download*

The entity is allowed to save a raw copy of the data and the entity will have no control over this copy neither log records regarding access to the copy. If Pure Log is adopted, the user's data will be directly downloadable in a pure form using a link. When an hyperlink is clicked and downloaded, the JAR file combines the data, decrypts and give it to the entity in raw form. In case of Access Logs, the entire JAR file will be given to the entity. Depending upon the entity the jar file can be accessed.

➢ *Timed access*

The view-only access works on the time basis and provides data for a limited period. The access starting time and duration are recorded by the pure log, meanwhile the Access Log is also utilised in a given time frame. By using the Network Time Protocol, the duration of access time is calculated. Based on the calculations, the time limit of Access Log records are determined. This timed access is compelled only it is combined with view access and download.

➢ *Location based access*

In this case, the Pure Log will record the location of the entities. The Access Log will verify the location for each of such access. The access is granted and the data are made available only to entities located at locations specified by the data owner.

*Dependability of Logs*

First, the intruder stores the JARs remotely to disturb auditing mechanism by, corrupting the JAR, or hides it to make the communication not possible for the user. Second, he gets the control of JRE that runs JAR files

*JARs Availability*

To protect offline JARs from the attackers, Cloud Information Accountability provides log harmonizer, which has two main tasks: dealing JAR copies and corrupt logs recovery.

Each log harmonizer maintains logger components copies holding similar data items. And the harmonizer being represented as a JAR file does not hold the user's data items for audition, but it allows client and server to communicate with logger components by providing the class file to them. Error correction information received from its logger components are stored by the harmonizer, and using the IBE, decryption key duplicate log records are found out from copies of the people's information JARs. As they are strictly combined with the logger component in a data JAR file, the user's data is copied along with the logger as it is strongly mixed with component. In sequence, the new copy of the logger also holds the old log records with respect to the usage of data in the original data JAR file. Because of old records redundancy is aroused and makes the data unfit for the new copies. The harmonizer merges copies from all log records and presents a clear view to the data owner by removing the redundancy.

*Log Correctness*

To maintain the log records correctly the JRE present in the logger component should not be modified. To check the logger component integrity, we depend on two-step process [1]:

1. We repair the JRE before the logger is launched and any kind of access is given, so as to provide guarantees of integrity of the JRE.
2. We insert hash codes, which calculate the hash values of the program traces of the modules being executed by the logger component. This helps us detect modifications of the JRE once the logger component has been launched, and are useful to verify if the original code flow of execution is altered.

## V. Conclusion

The system suggests some novel approaches for automatically logging any access to the data in the cloud together with an auditing mechanism. This mechanism allows the data owner to not only audit his content but also enforce strong back-end protection. Moreover, one of the main features of our work is that it enables the data owner to audit even those copies of its data that were made without this knowledge.

*Future work*

Apart from a class file for authenticating the servers or the users, another class file finds the correct inner JAR, a third class file which checks the JVM's validity using oblivious hashing.

## References Références Referencias

1. Ensuring Distributed Accountability for Data Sharing in the Cloud Author, Smitha Sundareswaran, Anna C.Squicciarini, Member, IEEE, and Dan Lin, IEEE Transactions on Dependable and Secure Computing ,VOL 9,NO,4 July/August 2012 .
2. P. Ammann and S. Jajodia, "Distributed Timestamp Generation in Planar Lattice Networks," ACM Trans. Computer Systems, vol. 11,pp. 205-225, Aug. 1993.

3. Sundareswaran, Anna C. Squicciarini, Member, IEEE, and Dan Lin "Ensuring Distributed Accountability for Data Sharing in the Cloud" Proc IEEE transactions on dependable and secure computing vol.9 no.4 year 2012.
4. Y. Chen et al., "Oblivious Hashing: A Stealthy Software Integrity Verification Primitive," Proc. Int'l Workshop Information Hiding, F. Petitcolas, ed., pp. 400-414, 2003.
5. B. Chun and A.C. Bavier, "Decentralized Trust Management and Accountability in Federated Systems," Proc. Ann. Hawaii Int'l Conf. System Sciences (HICSS), 2004.
6. Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, Song D. Provable Data Possession at Untrusted Stores. Proc. ACM Conf. Computer and Comm. Security 2007; 598- 609
7. OASIS Security Services Technical Committee, "Security Assertion Markup Language (saml) 2.0, "http://www.oasis-open.org/committees/tchome.php?wg abbrev=security, 2012.
8. R. Corin, S. Etalle, J.I. den Hartog, G. Lenzini, and I. Staicu, "A Logic for Auditing Accountability in Decentralized Systems," Proc. IFIP TC1 WG1.7 Workshop Formal Aspects in Security and Trust, pp. 187-201, 2005.
9. B. Crispo and G. Ruffo, "Reasoning about Accountability within Delegation," Proc. Third Int'l Conf. Information and Comm. Security (ICICS), pp. 251-260, 2001.
10. J. Hightower and G. Borriello, "Location Systems for UbiquitousComputing," Computer, vol. 34, no. 8, pp. 57-66, Aug. 2001.
11. E. Barka and A. Lakas, "Integrating Usage Control with SIP-BasedCommunications," J. Computer Systems, Networks, and Comm.,vol. 2008, pp. 1-8, 2008.
12. R. Jagadeesan, A. Jeffrey, C. Pitcher, and J. Riely, "Towards a Theory of Accountability and Audit," Proc. 14th European Conf. Research in Computer Security (ESORICS), pp. 152-167, 2009.
13. S. Pearson and A. Charlesworth, "Accountability as a WayForward for Privacy Protection in the Cloud," Proc. First Int'l Conf. Cloud Computing, 2009.
14. J.H. Lin, R.L. Geiger, R.R. Smith, A.W. Chan, and S. Wanchoo,Method for Authenticating a Java Archive (jar) for Portable Devices,US Patent 6,766,353, July 2004.
15. NTP: The Network Time Protocol, http://www.ntp.org/, 2012.
16. S. Pearson, Y. Shen, and M. Mowbray, "A Privacy Manager forCloud Computing," Proc. Int'l Conf. Cloud Computing (CloudCom), pp. 90-106, 2009.
17. J.W. Holford, W.J. Caelli, and A.W. Rhodes, "Using Self-Defending Objects to Develop Security Aware Applications in Java," Proc. 27th Australasian Conf. Computer Science, vol. 26,pp. 341-349, 2004.
18. R. Kailar, "Accountability in Electronic Commerce Protocols,"IEEE Trans. Software Eng., vol. 22, no. 5, pp. 313-328, May 1996.
19. D. Boneh and M.K. Franklin, "Identity-Based Encryption from theWeil Pairing," Proc. Int'l Cryptology Conf. Advances in Cryptology,pp. 213-229, 2001.
20. T. Mather, S. Kumaraswamy, and S. Latif, Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance (Theory in Practice), first ed. O' Reilly, 2009. M.C. Mont, S. Pearson, and P. Bramhall, "Towards AccountableManagement of Identity and Privacy: Sticky Policies and Enforceable
21. Tracing Services," Proc. Int'l WorkshopDatabase and Expert Systems Applications (DEXA), pp. 377-382, 2003.