Graduate Research in Engineering and Technology (GRET)

Volume 1 | Issue 2 Article 13

October 2013

ENCODING PERSONAL INFORMATION ON DATA SHARING IN CLOUD USING BASE64 ALGORITHM

P.L. RINI
Dept of CSE, M.N.M.Jain Engineering College, plrini@gmail.com

Y. GOLD ANAND.N
Dept of CSE, M.N.M.Jain Engineering College, yganandn@gmail.com

Recommended Citation
RINI, P.L. and ANAND.N, Y. GOLD (2013) "ENCODING PERSONAL INFORMATION ON DATA SHARING IN CLOUD USING BASE64 ALGORITHM," *Graduate Research in Engineering and Technology (GRET)*: Vol. 1 : Iss. 2 , Article 13.
DOI: 10.47893/GRET.2013.1031
Available at: https://www.interscience.in/gret/vol1/iss2/13

ENCODING PERSONAL INFORMATION ON DATA SHARING IN CLOUD USING BASE64 ALGORITHM

P.L.RINI[1], Y.GOLD ANAND.N[2]

[1]M.E Scholar, [2]Assistant Professor, Dept of CSE, M.N.M.Jain Engineering College

Abstract- A major feature of cloud services is that user data are processed remotely among machines. But user fears of losing control of their own data, particularly financial and health data can becomes a significant barrier to wide adoption of cloud services in order to avoid this problem we provide a novel approach, namely Cloud Information Accountability (CIA) for clients. So that the authorized client can only access the data in the cloud. Data owner store data in the format of JAR format thus client access data only by the permission of data owner. To strengthen user's control also provide a distributed audit mechanism by push and pull mode. Base64 encoding algorithm is used for encoding the JAR file in order to secure JAR file from attackers. Log maintained and send periodically to the data owner.

Keywords- *JAR File, Identity Based encryption (IBE), Base64Algorithm, Security, Encoding, Log creation, Audit, Authorized client, Cloud Information Accountability(CIA), Push and Pull mode.*

I. INTRODUCTION

Cloud computing is receiving a great deal of attention, both in publications and among users, from individuals at home to the U.S. government. Yet it is not always clearly defined. Cloud computing is a subscription-based service where you can obtain networked storage space and computer resources. One way to think of cloud computing is to consider your experience with email. Your email client, if it is Yahoo!, Gmail, Hotmail, and so on, takes care of housing all of the hardware and software necessary to support your personal email account. When you want to access your email you open your web browser, go to the email client, and log in. The most important part of the equation is having internet access. Your email is not housed on your physical computer; you access it through an internet connection, and you can access it anywhere. If you are on a trip, at work, or down the street getting coffee, you can check your email as long as you have access to the internet. Your email is different than software installed on your computer, such as a word processing program. When you create a document using word processing software, that document stays on the device you used to make it unless you physically move it. An email client is similar to how cloud computing works. Except instead of accessing just your email, you can choose what information you have access to within the cloud.

The CIA framework approach provides end-to-end accountability in a highly distributed fashion maintaining light-weight and powerful accountability that combines aspects of access control, usage control and authentication. Base64 Algorithm is used to encode the JAR file in order to provide security. Two auditing modes: push and pull mode developed. Push-Logs periodically send to data owner. Pull-Alternative approach where the user can retrieve the logs as needed. Supports a variety of security policies, like indexing policies for text files, usage control for executables, and generic accountability and provenance controls.

As the cloud provider gives the access of the resources to the cloud user's, the data owner keeps track of usage of their resources. It also provides authenticated usage of data in the cloud. It also handles copying, disassembling attack, man in the middle attack, compromised JVM attack. Cloud Information Accountability is being user here.

II. CLOUD INFORMATION ACCOUNTABILITY

The data flow in the CIA framework is as follows. At the beginning, each user creates a pair of public and private keys based on Identity-Based Encryption (IBE). Using the generated key, the user will create a logger (i.e., a JAR file) to store its data items, and sign and seal it. The JAR file includes a set of access control rules specifying whether and how the cloud servers and possibly other data stakeholders are authorized to access the data. Then, he/she sends the JAR file to the cloud service provider (CSP) that he/she subscribes to. To authenticate the CSP to the JAR, we use OpenSSL based certificates, wherein a trusted certificate authority certifies the CSP. In the event that the access is requested by a user, we employ SAML-based authentication, wherein a trusted identity provider issues certificates verifying the user's identity based on his username. Using SAML-based authentication for the users allows us to increase or decrease the number of users to whom the access is granted by simply adding more user identities and corresponding SAML certificates. Using OpenSSL for the CSP allows us to ensure that we grant access based on the role of the CSP. Thus, this design ensures that our architecture scales well both for users and CSPs.

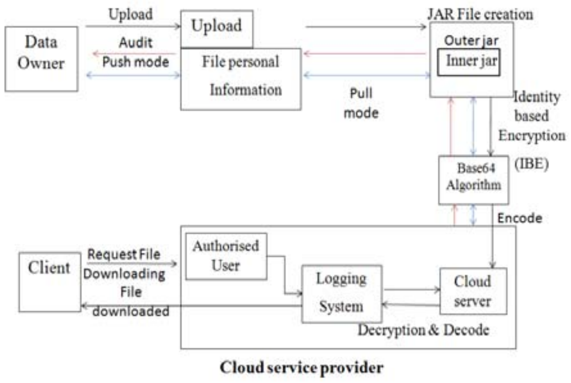


Fig. 1.Cloud Information Structure

Once the authentication succeeds, the CSP (or the user) will be allowed to access the data enclosed in the JAR. Depending on the configuration settings defined at the time of creation, the JAR will provide usage control associated with logging, or will provide only logging functionality. As for the logging, each time there is an access to the data; the JAR will automatically generate a log record, encrypt it using the public key distributed by the data owner, and store it along with the data. The encryption of the log file prevents unauthorized changes to the file by attackers. The data owner could opt to reuse the same key pair for all JARs or create different key pairs for separate JARs. Using separate keys can enhance the security without introducing any overhead except in the initialization phase. In addition, some error correction information will be sent to the log harmonizer to handle possible log file corruption.

The encrypted log files can later be decrypted and accessed by the data owner or other authorized stakeholders at any time for auditing purposes with the aid of the log harmonizer.

III. LOGGING MECHANISM

A logger component is a nested Java JAR file which stores a user's data items and corresponding log files. As shown in Figure 1, our proposed JAR file consists of one outer JAR enclosing one or more inner JARs.

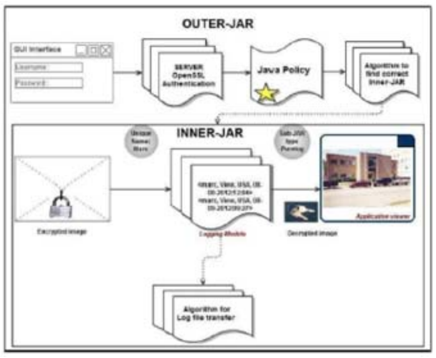


Fig. 2. The Structure of the JAR File

The main responsibility of the outer JAR is to handle authentication of entities which want to access the data stored in the JAR file. In our context, the data owners may not know the exact CSPs that are going to handle the data. Hence, authentication is specified according to the servers' functionality (which we assume to be known through a lookup service), rather than the server's URL or identity. For example a policy may state that Server X is allowed to download the data if it is a storage server. The outer JAR may also have the access control functionality to enforce the data owner's requirements, specified as Java policies, on the usage of the data. A Java policy specifies which permissions are available for a particular piece of code in a Java application environment 1. Moreover, the outer JAR is also in charge of selecting correct inner-JAR according to the identity of the entity who requests the data. Each inner JAR contains the encrypted data, class files to facilitate retrieval of log files and display enclosed data in a suitable format, and a log file for each encrypted item.

IV. END-TO-END AUDITING MECHANISM

The log harmonizer is responsible for auditing. Being the trusted component, the log harmonizer gen- erates the master key. It holds on to the decryption key for the IBE key pair, as it is responsible for decrypting the logs. Alternatively, the decryption can be carried out on the client end if the path between the log harmonizer and the client is not trusted. In this case, the harmonizer sends the key to the client in a secure key exchange. It supports two complementary auditing strategies: (i) push mode; (ii) pull mode.

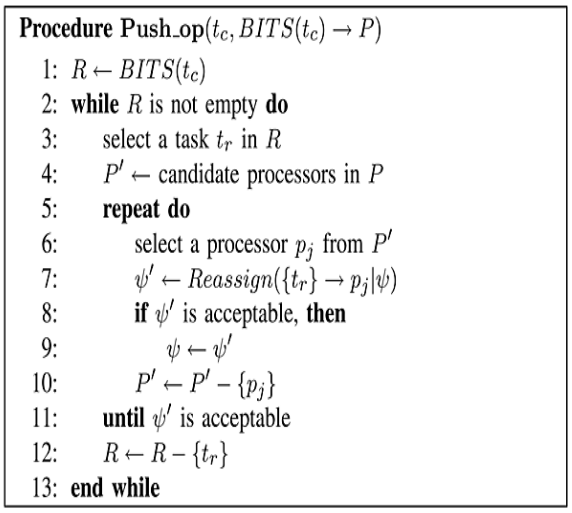
Push Mode: In this mode, the logs are periodically pushed to the data owner by the harmonizer. The push action will be triggered by either type of the following two events: one is that the time elapses for a certain period according to the temporal timer inserted as part of the JAR file; the other is that the JAR file exceeds the size stipulated by the content owner at the time of creation. After the logs are sent to the data owner, the log files will be dumped, so as to free the space for future access logs. Along with the log files, the error correcting information for those logs is also dumped.

This push mode is the basic mode which can be adopted by both the PureLog and the Access Log, regardless of whether there is a request from the data owner for the log files. This mode serves two essential functions in the logging architecture: 1) It ensures that the size of the log files does not explode and 2) It enables timely detection and correction of any loss or damage to the log files.

Pull Mode: This mode allows data owners to retrieve the logs anytime when they want to check the recent access to their own data. The pull message consists simply of an FTP pull command, which can be issues from the command line. For naive users, a wizard comprising a batch file can be easily built. The request will be sent to the harmonizer, and the user will be informed of the data's locations and obtain an integrated copy of the authentic and sealed log file.

ALGORITHM

The pushing mode may be preferred by data owners who are organizations and need 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. Further, as discussed in Section 7, supporting both pushing and pulling modes helps protecting from some non trivial attacks.



```
Procedure Push_op($t_c, BITS(t_c) \rightarrow P$)
1:  $R \leftarrow BITS(t_c)$
2:  while $R$ is not empty do
3:      select a task $t_r$ in $R$
4:      $P' \leftarrow$ candidate processors in $P$
5:      repeat do
6:          select a processor $p_j$ from $P'$
7:          $\psi' \leftarrow Reassign(\{t_r\} \rightarrow p_j|\psi)$
8:          if $\psi'$ is acceptable, then
9:              $\psi \leftarrow \psi'$
10:         $P' \leftarrow P' - \{p_j\}$
11:     until $\psi'$ is acceptable
12:     $R \leftarrow R - \{t_r\}$
13: end while
```

Fig 3.Push Algorithm

Our auditing mechanism has two main advantages. First, it guarantees a high level of availability of the logs. Second, the use of the harmonizer minimizes the amount of workload for human users in going through long log files sent by different copies of JAR files. For a better under- standing of the auditing mechanism.

V. BASE64 ALGORITHM

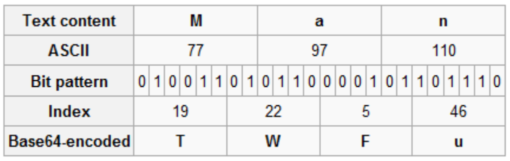
Base64 is a group of similar encoding schemes that represent binary data in an ASCII string format by translating it into a radix-64 representation. The Base64 term originates from a specific MIME content transfer encoding.Base64 encoding schemes are commonly used when there is a need to encode binary data that need to be stored and transferred over media that are designed to deal with textual data. This is to ensure that the data remain intact without modification during transport. Base64 is commonly used in a number of applications including email via MIME, and storing complex data in XML.

A. Applications

Base64 can be used in a variety of contexts:

Base64 can be used to transmit and store text that might otherwise cause delimiter collision.

- Base64 is used to store a password hash computed with crypt in the /etc/passwd.
- Spammers use Base64 to evade basic anti-spamming tools, which often do not decode Base64 and therefore cannot detect keywords in encoded messages.
- Base64 is used for PHP obfuscation.
- Base64 is used to encode character strings in LDIF files.
- Base64 is often used to embed binary data in an XML file, using a syntax similar to <data encoding="base64">…</data> e.g. favicons in Firefox's bookmarks.html.
- Base64 is used to encode binary files such as images within scripts, to avoid depending on external files.
- The data URI scheme can use Base64 to represent file contents. For instance, background images and fonts can be specified in a CSS style sheet file as data: URIs, instead of being supplied in separate files.



Text content	M								a								n							
ASCII	77								97								110							
Bit pattern	0	1	0	0	1	1	0	1	0	1	1	0	0	0	0	1	0	1	1	0	1	1	1	0
Index	19						22						5						46					
Base64-encoded	T						W						F						u					

Fig. 4. Base64 Character Values

THE BASE64 ENCODING PROCESS IS TO:

- Divide the input bytes stream into blocks of 3 bytes.
- Divide 24 bits of each 3-byte block into 4 groups of 6 bits.
- Map each group of 6 bits to 1 printable character, based on the 6-bit value using the Base64 character set map.
- If the last 3-byte block has only 1 byte of input data, pad 2 bytes of zero (\x0000). After encoding it as a normal block, override the last 2 characters with 2 equal signs (==), so the decoding process knows 2 bytes of zero were padded.
- Carriage return (\r) and new line (\n) are inserted into the output character stream. They will be ignored by the decoding process.

VI. RELATED WORK

To ensure data confidentiality, integrity, and availability (CIA) [1] the storage provider must offer capabilities that, at a minimum tested encryption schema to ensure that the shared storage environment safeguards all data;

Security and privacy Challenges in [2] for providing a trustworthy cloud computing environment.

Accountability increase trust in cloud computing.[4] ,is likely to become a core concept in both the cloud and in new mechanisms. CSPs have both legal and ethical obligations to ensure privacy and protect data and thereby demonstrate their services' trustworthy nature.

Protection mechanisms in [5] include encryption, marking data with different access levels to enable access control, and integrity verification. Privacy enabled Digital Rights Management Without Trusted Third Party Assumption [11] provides Digital rights management systems are required to provide security and accountability without violating the privacy. Privacy preserving content distribution mechanism for digital rights management without relying on the trusted third party assumption in order to avoid attack. The mechanism supports access control without degrading user's privacy.

Provable data possession (PDP) [16] that allows a client that has stored data at an untrusted server to verify that the server possesses the original data without retrieving it. The model generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs. The client maintains a constant amount of metadata to verify the proof. The challenge/response protocol transmits a small, constant amount of data, which minimizes network communication. Thus, the PDP model for remote data checking supports large data sets in widely-distributed storage systems.

VII. PROPOSED WORK

The CIA framework approach provides end-to-end accountability in a highly distributed fashion, combines aspects of access control, usage control and authentication.

The overall process can be divided into following operations.

- Cloud Information Structure
- JAR Creation and Security
- Logger Information Maintenance
- Data Outsourcing

A. Cloud Information Structure

Present an overview of the Cloud Information Accountability (CIA). The CIA framework conducts automated logging and distributed auditing at any cloud service provider. The overall CIA framework, users Registration, creating account and logging.

B. JAR Creation and Security

JAR files are created in order to reduce the Server Load and efficient transformation. Cloud servers and possibly other data stakeholders (users, companies) to be authorized to access the content itself. The JAR file sended to the cloud service provider (CSP).Access to the data will trigger an automated and authenticated logging mechanism local to the Jars.Base64 Algorithm used for Encoding and decoding JAR File.

C. Logger Information Maintenance

The programmable capability of JARs to conduct automated logging. Each log information has its own logging information. Since user's data is strongly coupled with the logger component in a data JAR file, the logger will be copied together with the user's data. The new copy of the logger contains the old log records with respect to the usage of data in the original data JAR file. Two auditing modes: push and pull mode developed.

Log records are generated by the logger component. Logging occurs at any access to the data in the JAR, and new log entries are appended sequentially.

D.Data Outsourcing

Our approach allows the data owner to not only audit his content but also enforce strong back-end protection if needed.

The main features of our work are that it enables the data owner to audit even those copies of its data that were made without his knowledge.

The User can access any file information available at any cause, also the user can avail the file information to any other users.

VIII. RESULT AND DISCUSSION

Pushing or pulling strategies have interesting tradeoffs. The pushing strategy is beneficial when there are a large number of accesses to the data within a short period of time. In this case, if the data are not pushed out frequently enough, the log file may become very large, which may increase cost of operations like copying data.

The pushing mode may be preferred by data owners who are organizations and need to keep track of the data usage consistently over time.

Base64 encoding algorithm is used for encoding the JAR file in order to secure JAR file from attackers. Log maintained and send periodically to the data owner.

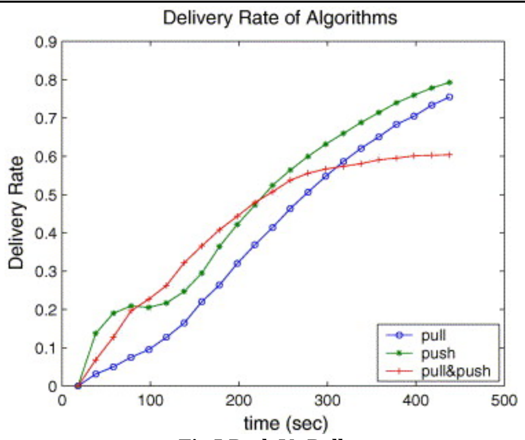


Fig 5.Push Vs Pull

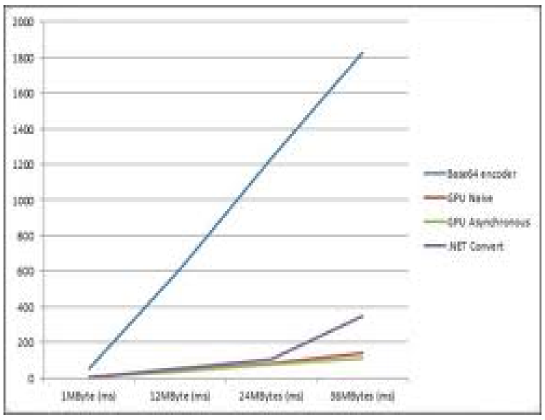


Fig 6. Base64 Algorithm Performance

CONCLUSION

Proposed innovative approaches for automatically logging and auditing mechanism. Our approach allows the data owner to not only audit his content but also enforce strong back-end protection if needed. The main features enables the data owners to audit even those copies were made without his knowledge. Eventhough encrypted, the encoding using Base64 Algorithm used to protect the data's from attackers.

REFERENCES

[1] Lori M.Kaufman, "Data Security in the World of Cloud Computing," Proc. IEEE Trans. Cloud Computing, 2009.

[2] Hassan Takabi and James B.D. Joshi and Gail-Joon Ahn, "Security and Privacy Challenges in Cloud Computing Environments," Proc. IEEE Trans. Cloud Computing, 2010.

[3] F. Martinelli and P. Mori, "On Usage Control for Grid Systems, "Future Generation Computer Systems, vol. 26, no. 7, pp. 1032-1042, 2010.

[4] Siani Pearson, "Toward Accountability in the Cloud," Proc IEEE Trans. Cloud Computing, 2010.

[5] Jianyong Chen, Yang Wang, and Xiaomin Wang, "On-Demand Security Architecture for Cloud Computing," Proc. IEEE Trans. Cloud Computing, 2012.

[6] 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, 2009.042,2010.

[7] S. Sundareswaran, A. Squicciarini, D. Lin, and S. Huang, "Promoting Distributed Accountability in the Cloud," Proc. IEEE Int'l Conf. Cloud Computing, 2011.

[8] Siani Pearson, (2010) "Toward Accountability in the Cloud," Proc.IEEE Trans. Cloud Computing.

[9] T. Mather, S. Kumaraswamy, and S. Latif,(2010) Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance (Theory in Practice), first. ed. O' Reilly, 2009.042.

[10] USENIX Association.J. Hightower and G. Borriello(2001). Location systems for ubiquitous computing. Computer, 34(8):57 –66.

[11] Lei Lei Win, Tony Thomas, and Sabu Emmanuel, "Privacy Enabled Digital Rights Management Without Trusted Third Party Assumption", Proc. IEEE Trans. Cloud Computing, 2012.

[12] R. Hasan, R. Sion, and M. Winslett, "The Case of the Fake Picasso: Preventing History Forgery with Secure Provenance," Proc.Seventh Conf. File and Storage Technologies, pp. 1-14, 2009.

[13] P.T. Jaeger, J. Lin, and J.M. Grimes, "Cloud Computing and Information Policy: Computing in a Policy Cloud?" J. Information Technology and Politics, vol. 5, no. 3, pp. 269-283, 2009.

[14] S.Pearson and A. Charlesworth, "Accountability as a Way Forward for Privacy Protection in the Cloud," Proc. First Int'l Conf. Cloud Computing, 2009.

[15] Lori M.Kaufman, "Data Security in the World of Cloud Computing," Proc. IEEE Trans. Cloud Computing, 2009.

[16] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson and Dawn Song "Provable Data Possession at Untrusted Stores", 14th ACM Conference on Computer and Communications Security (CCS 2007).

[17] Yuqun Chen[1], Ramarathnam Venkatesan[1], Matthew Cary[2], Ruoming Pang[3], Saurabh Sinha[2], and Mariusz H. Jakubowski[1] "Oblivious Hashing: A Stealthy Software Integrity Verification Primitive"

◈◈◈