Selective Encryption and component-Oriented Deduplication for Mobile Cloud Data Computing

[1]Uma H R, [2]Mrs. Shashikala S V
[1]P G Student, CS & E, BGSIT, Mandya
[2]Prof. & HOD,CS& E , BGSIT, Mandya-571418

Abstract—As smart devices gain their popularity and usage applications become versatile, the users are also hoping to perform resource intensive tasks at anywhere and anytime as conveniently as using their static computers. To overcome the smart device's intrinsic resource limitations in processing, storage, and power, emerging collaborative mobile cloud technologies such as Mobile Cloud Computing (MCC), Mobile-Edge Computing (MEC), and Fog Computing (FC) augment the smart device's capabilities by leveraging distributed and remote cloud resources. However, in collaborative computing environments, the demand for big data processing and exchanges among smart devices is considered as a significant challenge. An effective technique to reduce data at a source device is essential to save network bandwidth and storage spaces. It, in turn, improves the data processing overhead as well as reduces the security vulnerability caused by data movement among the smart devices.

In this paper, we design and develop a novel Selective Encryption and Component-Oriented Deduplication (SEACOD) application that achieves both fast and effective data encryption and reduction for MCC services. Specifically, SEACOD efficiently deduplicates redundant objects in files, emails, as well as images exploiting object-level components based on their structures. It also effectively reduces the overall encryption overhead on the mobile devices by adaptively applying compression and encryption methods according to the decomposed data types. Our evaluation using real datasets of structured files shows that the proposed scheme accomplishes as good of storage savings as a variable-block deduplication, while being as fast as a file-level or a large fixed-size block-level deduplication.

I. Introduction

As mobile devices become increasingly prevalent and the applications along with smart sensors/things become versatile, mobile services demand more resource intensive tasks and advanced interactivity for better Quality of Experience (QoE). To overcome a mobile device's intrinsic resource constraints such as CPU, storage, and battery life, Mobile Cloud Computing (MCC), Mobile-Edge Computing (MEC), and Fog Computing (FC) become emerging collaborative mobile cloud technologies that augment the mobile devices' capabilities by offloading towards resources in remote cloud computing platforms. As illustrated in Figure 1, the Mobile Cloud Service (MCS) model consists of *an agent-client infrastructure* and *acollaborative mobile cloud*. MCS includes numerous mobiledevices and sensors/things that are closely involved in various data intensive cloud activities under dynamic mobile environments. Considering the huge amounts of data (including pictures and videos) generated by smartphones, sensors, and things needing to be processed, moved, stored, and extracted over lower bandwidth and less reliable mobile connections, the

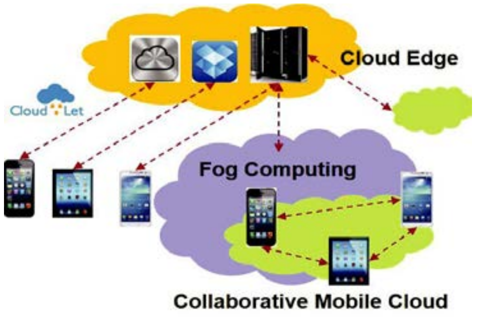


Fig. 1. A Mobile Cloud Service Model

MCS platform would require an efficient data centric facility and many instrumentations. Especially, efficient non server-side data reduction techniques are essential to save data on the path from a user to cloud servers or storage spaces. It, in turn, expedites the data processing and transmission speed as well as reduces mobile data vulnerability in the MCS platform. Although traditional server-side data deduplication techniques tend to achieve a high data reduction rate, as they require high processing overhead due to data chunking, index processing, and data fragmentation, they cannot be directly used in capacity limited mobile devices. While, a simple file-level or a large fixed-size block-level deduplication (i.e., Dropbox) may be able to cope with the limited source device capacity, it cannot produce a high data reduction rate.

In this paper, we propose a novel Selective Encryption and Component-Oriented Deduplication (SEACOD) application that *achieves both efficient and effective data deduplication* at a source site for MCS. SEACOD efficiently deduplicates redundant objects in structured files such as MS docx, pptx, and pdf by exploiting object-level components based on their structures, resulting to less data chunking overhead as well as fewer indexes than a block-level deduplication. It also reduces the overall encryption overhead as well as it can selectively apply efficient encryption methods according to the data types. SEACOD is effective in that its chunks are content-oriented objects and it does not have a boundary-shifting problem, thus achieving a higher data deduplication ratio than a file-level deduplication. The contributions of SEACOD are as follows:

(1) we have designed and implemented an efficient smartphone application to eliminate redundancies in structured files by

exploiting object-level components; (2) we have extensively evaluated the performance and overhead of SEACOD with real datasets of structured files. The evaluation results present that SEACOD achieves as good of storage savings as a variable-block deduplication, while being as fast as a file-level or a large (i.e., 4 MB as in Dropbox) fixed-size block-level deduplication; and (3) we have evaluated the performance of the encryption algorithms for many different file formats.

The rest of the chapter is organized as follows. Section II discusses the related work. We describe the SEACOD deduplication scheme and its implementation in Section III. We validate our approach in Section IV. We conclude the chapter in Section V.

II. Related Work

Many data chunking methods have been proposed to improve the performance of data deduplication. Traditional block-level deduplication [1] technologies chunck the data file into blocks of fixed or variable sizes. Since they achieve high deduplication rates by providing the fine granularity chunking techniques, it has been used for backup or file systems such as Venti [2] and Data Domain File System (DDFS) [3], as well as for removing redundant network traffic including Low Bandwidth File System (LBFS) [1]. However, as block-level deduplication techniques, especially variable-size ones, require the high cost of processing time and space (for example, the use of Rabin fingerprint matching [4]) and of maintaining and tracking large index and data fragmentation, it often runs on specialized fast and high-capacity servers for in-line or cloud storage systems. A client device of cloud-based storage is often limited in its processing capability and memory space to perform an effective traditional data deduplication.

Microsoft's Single Instance Server (SIS) [5] and EMC's Centera [6] employ a file-level deduplication. As it performs a simple chunking (a chunk is a file), it requires less index processing, and it has been used for many deduplication ap-plications with time and space limitations. For example, a data deduplication for in-line processing applications [7] uses it to cope with the costs of processing time and memory overheads. Many cloud-based storage services such as JustCloud [8], and Mozy [9] also employ single instance storage using a simple file-level deduplication. However, while it may be able to cope with the limited client device capacity, it cannot achieve a high data reduction rate.

Hybrid approaches use variable-size block-level dedupli-cation and file-level deduplication adaptively, according to the policy information. Dropbox [10] uses a large fixed size (4 MB chunk size) block-level deduplication in addition to single file instance storage. However, its data deduplication rate is still far less than a variable-size deduplication, due to the large granularity of chunks and the potential chunk boundary-shifting problem [11]. Min et al. [12] employ a context-aware chunking where they use a file-level dedupli-cation for multimedia content, compressed files, or encrypted content and use variable-size block-level deduplication for text files. The Hybrid Email Deduplication System (HEDS) [13]

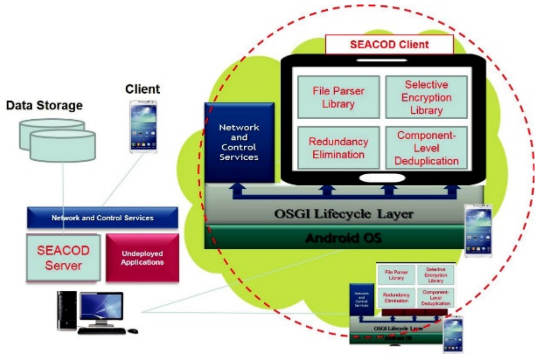


Fig. 2. SEACOD MCS Architecture

separates the email message body and individual attachments, and performs a variable-size block-level deduplication. HEDS uses a file-level deduplication for unstructured file types such as multimedia content and encrypted content. A few format-aware data deduplication techniques such as ADMAD [14], [15], SAFE [16], and [17], have been proposed to simplify the chunking mechanism by using the structured objects for the traditional server-based backup applications. Although the idea of format-aware data deduplication techniques to decompose a file into objects according to the object structure is similar to the proposed SEACOD approach, they are not designed as a client-based deduplication mechanism for mobile cloud systems.

III. Selective Encryption and Component-Oriented Deduplication (SEACOD) Architecture

In this section, we present the SEACOD MCS architecture, and explain the SEACOD algorithms as well as the decomposed object structures.

A. SEACOD Architecture

As illustrated in Figure 2, the SEACOD framework consists of a light-weight smartphone application (SEACOD client) and a server middleware (SEACOD server) on the Android smart-phone cloud platform. The SEACOD client is the main focus of this paper that consists of four modules including the file parser library, selective encryption library, component-level deduplication manager, and redundancy elimination protocol modules. An application with the four SEACOD client mod-ules is dynamically deployed and configured to each SEACOD client node via a network and control service interface. The ported OSGI lifecycle layer performs a wrapper to Android OS in support of the dynamic SEACOD client deployment. The network and control services create a library for task and membership control. Specifically, each SEACOD client module performs the following functionalities:

File parser library module: This includes various basic types of file parsers (i.e., docx, pptx, and pdfs) and image parsers as well as application aware file parsers (i.e., EHR XML files and images formatted by CCR, CDA, and

CCD). They decompose an original file into potentially many smaller sized objects according to the file structure policy. The file parser also can combine several small objects into a compound object based on the file parser's policy. It currently supports three file structure libraries including Microsoft Word (docx), Powerpoint (pptx), and an Adobe Portable Document (pdf). A PDF file defined at ISO 32000 [18] consists of a header, body, cross references, and a trailer. As shown in Figure 3, the header indicates the version of the pdf document. The body includes a series of objects. The cross reference has offsets of objects in the document, and the trailer has an offset of the cross reference section. MS docx and pptx files follow the Office Open XML format (called Open XML), which is standardized at ECMA-376 [19] and ISO/IEC 29500 [20]. As shown in Figure 4(a), texts of a Word file are contained in a document.xml object, and image objects are under a media directory, while other directories shown in the figure contain meta-data objects. Likewise, a Powerpoint file in Figure 4(b) has a media directory, but has different meta-data objects. In addition, texts per slide are structured into each individual slide<number>.xml. A presentation.xml holds the point-ers of the slide objects.

Selective encryption library module: This maintains the overall encryption overhead for different settings including different sizes of data blocks and different data types (text, images, and audio files) for each encryption algorithm. In order to SEACOD the overall encryption overhead including the battery consumption on the mo-bile devices, it selectively applies different encryption methods according to the decomposed data types and the packet sizes.

Component-level deduplication manager: This receives object indexes and decomposed objects of a file from the file parsers and checks the uniqueness of an index by checking the object index table. The decomposed objects are temporarily held in the object buffer. The component-level deduplication module checks the existence of ob-jects using the object index table, and a unique object is saved into storage.

Redundancy elimination protocol module: This up-loads the component indexes created by the SEACOD client's deduplication parser to the SEACOD server via a batch packet so that the SEACOD server can correlate the indexes to eliminate the redundant data exchange. When a SEACOD client needs to transfer data to the SEACOD server, it sends the component indexes first instead of the real data. After receiving the non-redundant component indexes from the SEACOD server, the SEACOD client transfers the non-redundant data components.

B. *SEACOD Algorithms*

SEACOD deduplication parser algorithm: As illus-trated in Figure 5, it parses the files into sub-file objects according to the provided file structure policies. If a file is

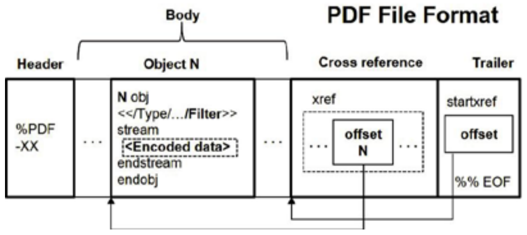
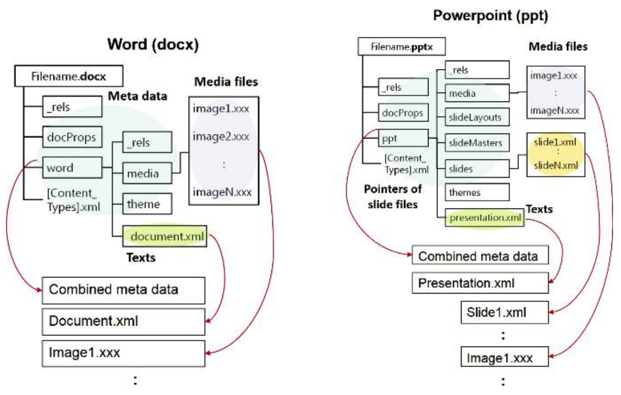


Fig. 3. Structure of a PDF file



(a) Word (docx) (b) Powerpoint (pptx)

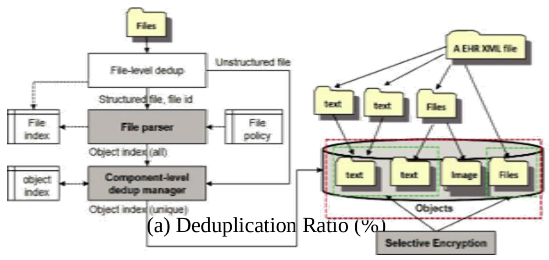
Fig. 4. Logical structure of MS office document file

one of the structured files supported by the file structure library, the structured file can be further parsed into the smaller file objects. If a file is an unstructured file type such as text files, a file object and its index will be saved into the storages without further parsing the file object.

The file parser extracts objects from a file according to the granularity configuration either of an object or a combination of objects. The file parser can combine several file objects to an object to reduce the number of object indexes. The component-level deduplication module performs an indexing process by using the object index table. Receiving the parsed object indexes of a file, the component deduplication module checks if each index is unique from the existing object index table. If a unique object is found, the object index will be saved into the object index table and the object data will be stored into a storage through the component manager as new data.

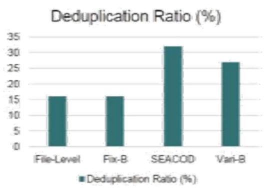
SEACOD redundant data elimination algorithm: It is network protocol to eliminate data redundancy. It saves bandwidth by exploiting commonality among compo-nents. First, when it writes files to the SEACOD server, it breaks files into many data components based on the file structure. It indexes the components by their hash value. Without sending that data over the network, it uploads indexes for the SEACOD server to look up the same data. It only sends non-redundant components.

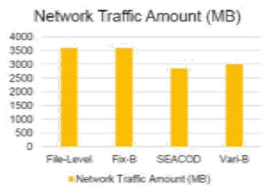
SEACOD selective encryption algorithm: If those unique components need to be stored in the remote



(a) Deduplication Ratio (%)

Fig. 5.SEACOD MCC Algorithms





(b) Network Traffic Amount (MB)

Fig. 6. SEACOD Performance

storage, the selective encryption module chooses the encryption algorithm according to the data sizes and systems. We initially use the performance evaluation of selected symmetric encryption algorithms such as AES, DES, and 3DES, RC6, Blowfish, and RC2 on power consumption for wireless devices.

IV. Evaluations

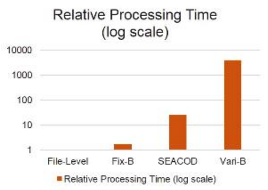
We compare the performance of SEACOD with a file-level deduplication scheme, a fixed-size block-level deduplication scheme, and a variable-size block-level deduplication scheme in the aspects of the deduplication ratio and network traffic amount. The deduplication ratio indicates how much storage space can be saved by removing redundancies and is computed
by Equation (1).

$$\left(\frac{InputDataSizeConsumedStorageSize}{InputDataSize}\right) 100 \quad (1)$$

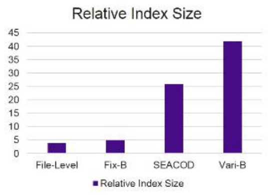
The network traffic amount indicates how much non-redundant data are transferred to the remote storage. We also measure the overhead metrics in the aspect of the processing time and index size. Since the overhead is proportional to the data size, we compare the relative processing time and index size overhead to the file-level, fixed-size block-level, and variable-size block-level deduplication schemes. For the variable-size block-level deduplication, we use 2 KB, 8 KB, and 64 KB as minimum, average, and maximum chunk sizes, respectively. For fixed-size block-level deduplication, we use 4 MB as the fixed block size as Dropbox does. Fixed-size block-level deduplication thus is the same as the file-level deduplication for files smaller than 4 MB. We carried out the evaluations on Fedora 16 Linux operating systems of kernel 2.6.35.9 SMP on Intel Core 2 Duo 3GHz.

A. Performance Evaluations

We first present experimental results of performance savings in SEACOD compared to the existing techniques. For dedupli-cation ratio as shown in Figure 6(a), SEACOD achieves about a 40% better performance than the file-level deduplication approach. SEACOD also shows a 15% better performance than the variable-size block-level deduplication that is the best deduplication ratio among the existing techniques. It is because SEACOD can find the object boundaries more efficiently than the variable-size block-level deduplication. For network traffic amount in Figure 6(b), SEACOD shows the lowest data traffic followed by the variable-size block-level deduplication.



(a) Processing Time Overhead



(b) Index Size Overhead

Fig. 7. SEACOD Overhead

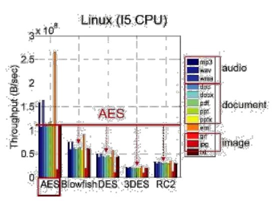
B. Overhead Evaluations

We now compare the overhead for the different deduplica-tion topologies as shown in Figure 7. The processing time overhead in Figure 7(a) presents that the fixed-size block-level and file-level deduplication schemes are the fastest. Although SEACOD takes a little longer than the file-level deduplication, SEACOD is faster by two orders of magnitudes than the variable-size block-level deduplication. The memory size increases proportionally to the number of indexes as shown in Figure 7(b). SEACOD has more index size than the fixed-size block-level and file-level deduplication schemes, but SEACOD incurs 2 times less index overhead than the variable-size block-level deduplication.

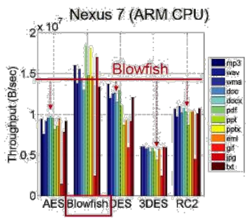
C. Selective Encryption Evaluations

We have tested how encryption algorithms perform ac-cording to the different file types and systems (i.e., with different CPU types). In Figure 8, we observed that encryption algorithms perform differently according to the CPU types. As shown in Figure 8(a), we measured the throughput of various encryption algorithms (AES, Blowfish, DES, 3DES, and RC2) that run on both Intel I5 based Linux and ARM based Nexus 7 devices for the different file types such as audio, document, and image files. The results show that AES outperforms other encryption algorithms in the Intel I5 based Linux device, but blowfish is the highest performance on the ARM based Nexus 7 device in Figure 8(b). The reason for AES's high performance in an Intel-based Linux system is that the Intel CPU has hardware-support instruction sets for AES encryption called an AES new Instruction (AES NI).

We also evaluated the processing time of the encryption algorithms (AES, Blowfish, DES, 3DES, and RC2) that run on both Intel I5 based Linux and ARM based Nexus 7 devices for the different data sizes from 4 KB to 100 MB. Overall,

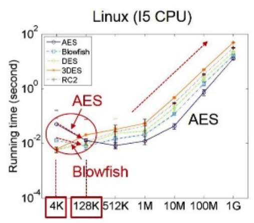


(a) Linux (I5 CPU)

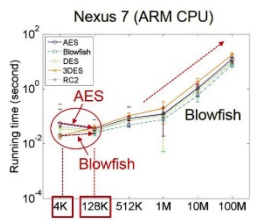


(b) Nexus 7 (ARM CPU)

Fig. 8. Throughput per System



(a) Linux (I5 CPU)



(b) Nexus 7 (ARM CPU)

Fig. 9. Time per File Size

as shown in Figure 9(a) and 9(b), processing time increases proportionally to data size. However, the processing time from 4 KB to 128 KB decreases for both AES and blowfish in both systems. The results indicate the importance of choosing the right deduplication granularity. As a result, we need to select a granularity maintaining a balance between removing redundancies and encryption processing time.

Overall, AES's performance is better than other encryption algorithms on an Intel-based system. However, for ARM-based smart devices, blowfish shows the best performance among the encryption algorithms.

V. Conclusion

We have presented a novel Selective Encryption and Component-Oriented Deduplication (SEACOD) application that achieves effective data reduction, efficient encryption, and data-oriented collaboration control for resource intensive mission-oriented mobile cloud computing services. Specif-ically, (1) we built an effective software framework for smartphones to eliminate redundancies in structured files by exploiting object-level components; (2) we designed efficient methods to reduce the overall encryption overhead on the mobile devices by selectively applying encryption methods ac-cording to the decomposed data types; and (3) we developed an intelligent mechanism to avoid the unnecessary data exchanges by exploring the collaborating members' data processing and transfer capability and existing data components.

As for future work, we plan to extend our prototype system to incorporate with the image and video files as well as support other Electronic Health Record (EHR) files such as a DICOM format. We also want to develop an intelligent mechanism to avoid the unnecessary data exchanges by exploring the collaborating members' data processing and transfer capability
and existing data components. We will further identify the different encryption methods and procedures for the SEACOD
framework.

References

[1] A. Muthitacharoen, B. Chen, and D. Mazieres, "A low-bandwidth network file system," in *SOSP '01 Proceedings of the eighteenth ACMsymposium on Operating systems principles*, vol. 35. ACM, Dec. 2001,pp. 174–187.

[2] S. Quinlan and S. Dorward, "Venti: A new approach to archival storage," in *Proceeding of the USENIX Conference on File and StorageTechnologies(FAST)*, vol. 4, Jan. 2002.

[3] B. Zhu, K. Li, and H. Patterson, "Avoiding the disk bottleneck in the data domain deduplication file system," in *Proceeding of the USENIXConference on File and Stroage Technologies(FAST)*, vol. 18, 2008.

[4] M. O. Rabin, "Fingerprinting by random polynomials," Harvard Univer-sity, Tech. Rep. Report TR-15-81, 1981.

[5] W. Bolosky, S. Corbin, D. Goebel, and J. Douceur, "Single instance storage in windows 2000," in *Proceedings of the 4th USENIX WindowsSystems Symposium*. USENIX, 2000, pp. 13–24.

[6] EMC, "Centera: Content addresses storage system, data sheet," http://www.emc.com/collateral/hardware/data-sheet/c931-emc-centera-cas-ds.pdf.

[7] D. T. Meyer and W. J. Bolosky, "A study of practical deduplication," in *Proceeding of the USENIX Conference on File and Storage Technolo-gies(FAST)*. USENIX, Feb. 2011.

[8] JustCloud, http://www.justcloud.com/.

[9] Mozy, http://mozy.com/.

[10] Dropbox, http://www.dropbox.com.

[11] K. Eshghi and H. K. Tang, "A framework for analyzing and improving content-based chunking algorithms," Hewlett-Packard Labs, Tech. Rep. HPL-2005-30(R.1), 2005.

[12] J. Min, D. Yoon, and Y. Won, "Efficient deduplication techniques for modern backup operation," *IEEE Transactions on Computers*, vol. 60, no. 6, pp. 824–840, June 2011.

[13] D. Kim and B.-Y. Choi, "Heds: Hybrid deduplication approach for email servers," in *Ubiquitous and Future Networks (ICUFN), 2012 FourthInternational Conference on*. IEEE, 2012, pp. 97–102.

[14] C. Liu, Y. Lu, C. Shi, G. Lu, D. Du, and D. Wang, "Admad: Application-driven metadata aware de-duplication archival storage system," in *Stor-age Network Architecture and Parallel I/Os, SNAPI'08. Fifth IEEE International Workshop on*. IEEE, 2008, pp. 29–35.

[15] J. Li, L.-w. He, S. Sengupta, and A. Aiyer, *MULTIMODAL OBJECTDE-DUPLICATION*, Microsoft Corporation, 08 2009, patent.

[16] D. Kim, B.-Y. Choi, and s. Song, "Safe: Structure-aware file and email deduplication for cloud-based storage systems," in *Cloud Networking(CloudNet), 2013 Second International Conference on*. IEEE, 2013.

[17] F. Yan and Y. Tan, "A method of object-based de-duplication," *Journalof Networks*, vol. 6, no. 12, pp. 1705–1712, 2011.

[18] Adobe, "Iso32000 : Document management - portable document format," http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/ pdfs/PDF32000 2008.pdf.

[19] E. C. M. A. (ECMA), "Standard ecma-376 : Office open xml file for-mats," http://www.ecma-international.org/publications/standards/Ecma-376.htm.

[20] T. I. O. for Standard Organization (ISO) and the Interna-tional Electrotechnical commission (IEC), "Iso/iec 29500-1:2008," http://www.iso.org/iso/iso catalogue/catalogue tc/ catalogue detail.htm?csnumber=51463.