Energy-Efficient Incremental Integrity for Securing Storage in Mobile Cloud Computing

Wassim Itani Ayman Kayssi Ali Chehab
Department of Electrical and Computer Engineering
American University of Beirut
Beirut, Lebanon
{wgi01, ayman, chehab}@aub.edu.lb

Abstract— **We present an energy-efficient protocol for ensuring the integrity of storage services in mobile cloud computing. The proposed protocol applies the concepts of incremental cryptography and trusted computing to design secure integrity data structures that protect the customer data while highly reducing the mobile client energy consumption and efficiently supporting dynamic data operations. The system design is analytically analyzed and experimentally implemented to demonstrate the energy savings it provides on mobile clients.**

I. INTRODUCTION

Although the adoption of cloud computing services is realizing rapid proliferation, customers of critical cloud services are reluctant and skeptical about the integrity of their stored data in the remote cloud. This fact is further aggravated in mobile cloud computing models that support battery-operated wireless and mobile client devices with limited energy resources. Any integrity enforcement mechanism should securely support dynamic operations on remote data and consider the limitations of mobile customers by employing energy-efficient algorithmic techniques and cryptographic data structures.

We present an energy-efficient protocol for ensuring the integrity of storage services in mobile cloud computing. The proposed protocol applies the concepts of incremental cryptography and trusted computing to design secure integrity data structures that protect the customer documents while highly reducing the mobile client energy consumption and efficiently supporting dynamic data operations. The system design is analytically analyzed and experimentally implemented to demonstrate the energy savings it provides on mobile clients.

II. RELATED WORK

The integrity protocol we present employs the incremental cryptography primitives developed by Bellare, Goldreich, and Goldwasser in [1, 2]. The main idea behind their work is to design a set of hash and Message Authentication Code (MAC) functions that support the incremental update property. This means that if a message M with a MAC MAC_M is updated by inserting/deleting a block of data, the incremental function can securely generate an updated MAC value using MAC_M and the inserted/deleted block without the need of regenerating the MAC value on all the message contents from scratch. The trusted computing model is based on the cryptographic coprocessor applications of the Dyad project [3].

III. SYSTEM DESIGN

The system model consists of three main entities: a mobile client that consumes the cloud storage services, a cloud service provider that manages and operates the cloud storage facility, and a trusted third party that configures and distributes tamperproof cryptographic coprocessors for installation in the remote cloud. Each crypto coprocessor is allocated to multiple registered clients and shares with each individual client a unique shared secret K_s. The system operation is divided into three phases: the initialization phase, the data update phase, and the integrity verification phase. A schematic diagram detailing the interaction among the cloud entities in each protocol phase is presented in Figure 1.

The initialization phase: In this phase the mobile data is prepared with the incremental authentication codes before migration to the cloud. For every client file F_x marked for cloud migration, an incremental MAC MAC_{Fx} is created using the shared key K_s. The generated incremental MACs are stored locally on the mobile client side and the corresponding files are transferred to the cloud storage.

The data update phase: This phase demonstrates how the incremental cryptographic concepts can be utilized for securely and efficiently supporting dynamic operations on the cloud data. Three main dynamic operations are adopted: File creation, file block insertion, and file block deletion. Note that the insertion and deletion operations form the building block for other file operations such as block replacement and move.

1. File creation: The file creation steps are analogous to those followed in the initialization phase. To secure the integrity of a newly created file F_{n+1}, the mobile client generates an incremental MAC on this file using the shared secret K_s. $MAC_{Fn+1}$is stored locally while $F_{n+1}$is transferred to the cloud for remote storage.
2. File block insertion: To insert a block B at position i in file F_x, the mobile client checks the availability of the file in local storage before applying the integrity operations. If F_x is not available, the mobile client requests it from the remote cloud storage. The cloud service provider processes the file request by retrieving the file from storage and sending a copy to the mobile client as well as to the trusted crypto coprocessor. Upon receiving F_x, the trusted coprocessor generates the incremental MAC MAC'_{Fx} using the shared key K_s and sends this MAC value to the mobile client. Once the mobile client receives the file contents and the corresponding MAC it checks the integrity of the file by comparing the stored MAC_{Fx} with the received MAC'_{Fx}. If the two MACs are equal, the mobile client proceeds to the file insertion operation as follows: it inserts block B at position i in file F_x and updates MAC_{Fx} by applying the incremental MAC operation on the inserted block only using the old MAC value and the key K_s. The insertion operation ends by sending the inserted block B to the remote cloud storage for synchronization.
3. File block deletion: To delete the block at position i in file F_x, the deletion operation follows similar steps to the insertion operation. The only change is that the MAC update in this case depends on the deleted block and the old MAC.

The data verification phase: In this phase the mobile client can request the integrity verification of a file, collection of files, or the whole file system stored in the remote cloud. To avoid

978-1-4244-8274-0/10/$26.00 ©2010 IEEE

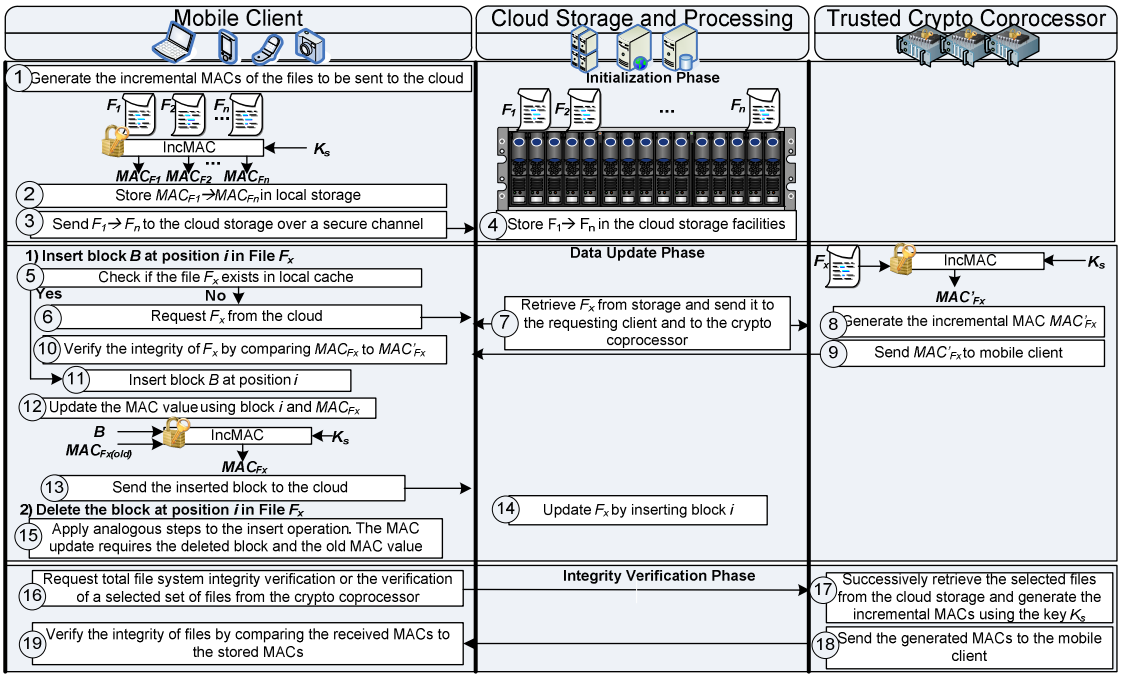

Figure. 1. System design and protocol interaction among the mobile client, cloud provider, and the crypto coprocessor

expensive data transfer operations to retrieve the files to be verified from the remote cloud storage and to reduce the processing overhead resulting from client-side integrity verification, the core integrity checking mechanism is offloaded to the crypto coprocessor. This phase starts by the mobile client requesting the integrity verification of a set of files or the whole file system from the trusted crypto coprocessor. The trusted coprocessor successively retrieves the selected files from the cloud storage, generates their incremental MACs using the shared secret K_s, and sends the generated values to the mobile client. The mobile client can verify the integrity of files by applying cheap comparison operations between the stored and received MAC values. The MAC generation by the crypto coprocessor and the incremental MAC update mechanism in the data update phase contributes to major savings in energy consumption on the mobile client. Let E_{IMac} be the energy consumed by the incremental MAC function when processing 1 byte of data on the mobile client and S_{Fx} be the size of F_x in bytes. The energy savings per file update, $E_{MAC-Gen}$, due to the crypto coprocessor MAC generation design choice is given as follows: $E_{MAC-Gen} = E_{IMac} \times S_{Fx}$.To calculate the energy savings due to the incremental MAC update mechanism, we need to subtract the energy cost incurred by the incremental MAC update operation from that incurred by the traditional approach using the well-known CBC MAC function to generate a MAC on the whole file contents. Let E_{CBC} be the energy consumed by the CBC MAC algorithm when processing 1 byte of data and M_{Fx} be the percent data modification applied on F_x due to block insertion or deletion. Since $E_{IMac} = 1.05 \times E_{CBC}$ [1], then the energy savings per file update E_{update} is given as follows: $E_{update} = S_{Fx} \times E_{IMac} \times (1/1.05 - M_{Fx}) \approx S_{Fx} \times E_{IMac} \times (1 - M_{Fx})$. The energy savings increase as M_{Fx} decreases since the incremental function is processing fewer bytes compared to the total file size. The energy savings in the verification phase has two main components: a processing component and a network reception component. Let $S_{F1 \to Fr}$ be the combined size of the files to be verified by the crypto coprocessor in bytes and $E_{Net-recv}$ be the energy consumed by the mobile client transceiver upon receiving 1 byte of data. The total application-layer energy savings in the verification phase on the mobile client due to delegating the MAC generation to the crypto coprocessor is given as follows: $E_{verify} = S_{F1 \to Fr} \times (E_{IMac} + E_{Net-recv})$.

IV. System Implementation

The system design was experimentally tested by a proof of concept implementation using an emulated cloud computing unit. The virtualization layer is provided using VMware Workstation 7.1. The mobile client is an HP iPAQ Pocket PC device running Windows Mobile 5 with a 300MHz processor and 64 MB of RAM. We evaluated the energy savings by executing a set of fifty typical insertion and deletion commands (add/delete a line of text, add/delete a paragraph, cut/paste lines and paragraphs, add/remove an image, etc.) on thirty MS Office Word documents. The file sizes range from 100 KB to 2 MB. The average execution time of the incremental MAC generation on the mobile device was found to be 37 ms compared to 420 ms of the traditional integrity approach that generates the file CBC MAC from scratch. This is over 90% reduction in processing requirements and thus in energy consumption on the mobile device. The processing and energy savings are proportional in this case, since the incremental MAC and the CBC MAC functions employ analogous system operations.

V. Conclusion

This paper presented an energy-efficient security protocol for ensuring storage integrity in mobile cloud computing. We provided a brief analytical analysis and an experimental proof of concept to demonstrate the energy savings realized.

Acknowledgment

This work was supported by Intel's Middle East Energy Efficiency (MER) Research Program.

References

[1] M. Bellare, O. Goldreich, and S. Goldwasser,*Proc. 27th Symposium on the Theory of Computing*, pp. 45-56, 1995.

[2] M. Bellare, O. Goldreich, and S. Goldwasser, *Crypto '94*, Vol 839, Springer-Verlag, pp. 216-233, 1994.

[3] C J. D. Tygar and B. Yee, In Proc. of *IP Workshop*, 1994.