Empirical Study of Accelerating Data Protection for Multi-tenant Storage

[1] Shu Qin Ren, [2]Shu Hao Zhang, [3]Yong Zhen Chen, [1]Miguel Rodel Felipe, [3]Ya Jun Ha, [1]Khin Mi Mi Aung
*[1]*Data Storage Institute, Singapore {REN_Shuqin, Mi_Mi_AUNG}@dsi.a-star.edu.sg*
[2]Nanyang Technology University, Singapore SHZHANG1@e.ntu.edu.sg@ntu.edu.sg
[3]National University of Singapore elecyz@nus.edu.sg@nus.edu.sg

Abstract

Multi-tenant shared storage systems are foundation for the deployment of cloud storage service. With the popularity of cloud storage service, data protection in this multi-resource and multi-tenant environment becomes even more critical and challenging, among which performance degrading, operation complexity and scalability resulted from data protection using cryptal are the hottest issue. This paper proposed a high-throughput multi-tenant data protection over distributed storage system, providing low-penalty data protection mechanism for multi-tenant cloud storage with good scalability and transparent setup of data protection engine. The contribution of this paper is threefold: 1) High-throughput cipher process by offloading to hardware and speeding up with pipelining; 2) Transparent data protection at block level for multiple tenants; 3) Plug-in data protection solution to support cloud storage.

Keywords: *Privacy protection on cloud storage, Data protection acceleration, High-throughput ciphering*

1. Introduction

Multi-tenant shared storage systems are foundation for the deployment of cloud storage service. With the popularity of cloud storage service, data privacy and security issues with the corresponding performance overhead introduced, operation complexity and scalability support are highly addressed. Data protection simply by encryption mechanism introduces computation overhead and latency to the overall performance. Secure, transparent yet light overhead data protection solution is highly demanded as a middle ware on cloud storage.

Threats in a large scale shared storage system can be classified as three aspects: 1) guest-to-cloud threats to attack storage service; 2) guest-to-guest threats to peep or violate the neighbour guests' data; 3) Cloud-to-guest threat to peep or violate the guests' data by taking advantage of data holder. With these threats, data isolation and privacy preserving are key challenges for large scale shared storage systems where multiple tenants are sharing multiple resources.

There are mainly two architectures proposed against these attacks on cloud storage. one is to enforce security mechanism such as encryption, access control on service provider, the other is on clients' site. The former architecture is based on the assumption of trusted cloud storage service which is not acceptable. The later architecture introduces operation complexity to end users and contradicts the advantages of cloud storage.

Based on the attacks existing on cloud storage with multi-resources and multi-tenants, we have been working on efficient data protection and access control mechanism by keeping the scalability, convenience and utilization of cloud storage service. We proposed a collaborative architecture to let data owner (client) and data holder (service provider) to work together to defend the three types of attacks mentioned above. Since cloud storage service provider has powerful yet cheaper facilities, data isolation and access control are enforced on storage service, but the data owner can still hide the exact policies and let storage provider to execute the access control. For protection on data itself, we proposed a secure gateway for each organization to encrypt/decrypt data put/get to/from cloud storage while keeping transparent operation from end user.

This paper is organized as following, Section 2 describes preliminary on device mapper, asynchronous block cipher driver and related performance issues; Section 3 illustrates the accelerated data protection processes for shared storage through optimized cipher driver, pipelined processing and multi-channel DMA; Section 4 shows the experiment results by comparing the performance on only software, default driver with FPGA and accelerated version; section 5 ends with some works to do in future.

In this paper, we demonstrate ***how to protect data privacy atop of today's cloud storage with less performance penalty***. The scheme provides the following special attributes:

- Secure outsourced data management. The encryption keys are managed by organizer itself based on fine-grained policies. The encrypted data are managed by third cloud storage providers.
- The schemes ensure data owner privacy protection while leaving data management to the service provider.
- The schemes ensure high throughput and transparent data protection between data owner and service provider.

2. Preliminary Works

2.1 From device mapper to device driver to FPGA

The transparent data protection workflow can be completed through device mapper, such as dm-crypt [4,5]. Dm-crypt is a transparent disk encryption subsystem. It is part of the device mapper infrastructure, and uses cryptographic routines from the kernel's Crypto API. dm-crypt is implemented as a device mapper target and may be stacked on top of other device mapper transformations. It can thus encrypt whole disks (including removable-media), partitions, software RAID volumes, logical volumes, as well as files. It appears as a block device, which can be used to back file systems, swap or an LVM physical volume.

The dm-crypt device mapper target resides entirely in kernel space, and is only concerned with encryption of the block device - it does not interpret any data itself. It relies on user space front-ends to create and activate encrypted volumes, and manage authentication. Such front-ends currently available include cryptsetup and cryptmount. We choose cryptsetup as our front end which is used to conveniently setup dm-crypt managed device-mapper mappings. LUKS [5] for dm-crypt is implemented in an enhanced version of cryptsetup. LUKS is the standard for Linux hard disk encryption. By providing a standard on-disk-format, it does not only facilitate compatibility among distributions, but also provides secure management of multiple user passwords. In contrast to existing solution, LUKS stores all setup necessary setup information in the partition header, enabling the user to transport or migrate his data seamlessly, based on above reason, we use cryptsetup-luks extension as the project tool.

Device-mapper crypt essentially target provides transparent encryption of block devices using the kernel crypto API, however due to the limitation of software performance, we are proposing a way of shifting the crypt work to high speed, fully pipelined FPGA card. To connect the dmcrypt device mapper with FPGA device, new device driver is designed and implemented.

2.2 Performance Related Issues

- **Data Transaction and Transfer**. Programmed input/output (PIO) vs. Direct Memory Access (DMA). PIO modes require a great deal of CPU overhead to configure a data transaction and transfer the data compared with DMA. System performance is heavily affected by PIO. To maximize the efficiency of DMA, PIO instructions are avoided if possible.
- **Crypto Block Driver.** Synchronous block driver vs. Asynchronous Driver [texbookCrypto]. Synchronous block driver makes cipher process block by block serially, which decrease the chunk data throughput. Asynchronous block driver could let FPGA based cryptographic engine pipelining with the device mapper to enhance total throughput tremendously.
- **Multiple DMA Channels.** With the data direction to and from FPGA device, at least two channels for each transaction could exploit the duplex nature of the PCIe link.

3. High-throughput Data Protection for Multi-tenant Storage

3.1 System Architecture

This system is to protect data for multi-tenant cloud storage where data flow from multiple computing servers at different organizations. A collaborative solution is deployed to ensure secure and high performance cloud storage. Two security aspects are to be addressed and enforced in a server-client collaborative way: 1) Secure Data Asset Management on Large Scale Shared Storage System to emphasize secure data access control; 2) Accelerating data protection for Large shared storage to emphasize high performance data crypt processing.

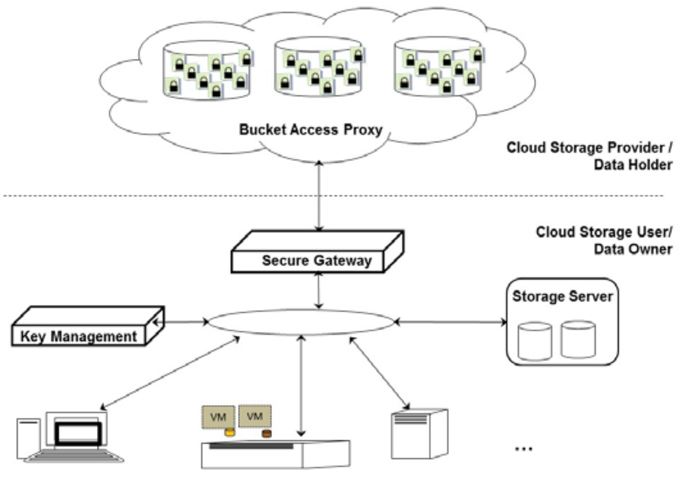

Figure 1. Data Protection System for Multi-tenant Storage

3.2 Secure Data Asset Management on Large Scale Shared Storage System

Since the cloud storage provider is the data holder for multiple clients, data isolation and access control executed on service site would be efficient and feasibility to both clients and service provider. With data isolation and access control [7,8,9,10], guest-to-guest attack and guest-to-cloud attack can be eliminated. Still, the enterprise doesn't want to release its access control policies to the service provider. How can service provider provide a secure data access control satisfied by clients? We applied attributes based access control with our previous work on privacy preserved key words searching [1]. By this, enterprise can hide its original attributes and set the encrypted attributes based access policies on cloud storage side. And the storage provider can implement and deploy access control on these privacy preserved attributes. The dataflow can refer figure 2 with step 1 to 5: 1) Organization A set access policy based on some attributes; 2) These attributes will be encrypted into meaningless attributes A_1, A_2 and so on; 3) based on these encrypted attributes, the corresponding access policy can be set on storage service provider who doesn't know the original attributes but still can assess whether that attribute meet or not; 4) The Policy Manager deploy its privacy preserved attributes to its staffs; 5) When staffs from the enterprise access storage, storage server has access proxy to check its corresponding privilege.

3.3 Accelerating data protection for Large shared storage

Besides secure data isolation and access control, we also investigated to accelerate data protection procedure to eliminate the latency brought in and minimize the operation complexity. Since data means value to enterprise, data protection by encryption has to be done before delivering to the cloud. To remove the operation complexity, we propose a secure gateway to do data protection for the enterprise data. Every staff's data will be encrypted automatically before going to the storage cloud; it is completely transparent to the staff without any operation enforced on the staff's computer. However this centralized mechanism brings in I/O performance and the corresponding latency issue. To eliminate the bottleneck and performance penalties caused from data encryption, a series of accelerating processes are deployed on the secure gateway. By incorporating hardware crypt task and pipelining process with software, the hardware performance is nearly made full usage of to speed up 4 times compared to the traditional mechanism. We also built a prototype to test out the practicality and feasibility. The positive results show its obvious improvement on big throughput and low latency support.

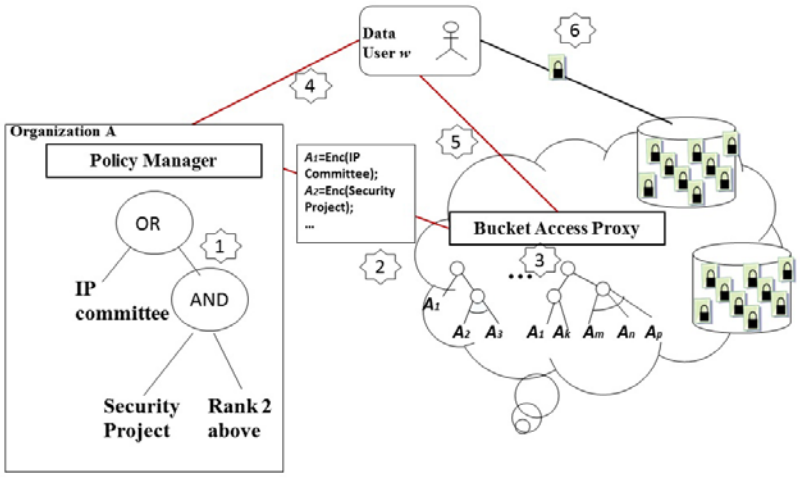

Figure 2. Dataflow of Secure Data Access Control

3.4 Asynchronous Block Driver Design And Setup

As mentioned in Preliminary Works, with the poor performance of crypto API, new device driver is desired to connect device mapper and FPGA device to give high speed cipher process. To eliminate the performance penalty caused by protection, FPGA based encryption engine is designed and implemented. Since the cipher process shifted to FPGA, minimizing the cost of data transfer data between CPU and FPGA has tremendous influence on the performance. For the default synchronize block cipher, request is sending from device mapper with all information extracted and cipher process is done on each 64 Bytes serially, introducing high delay and communication cost. To enhance the performance, bigger block transfer and asynchronous block cipher are new designed and implemented.

However for an asynchronous request is only a pointer to a request structure managed by device mapper cryptsetup, thus it must first extract the data to cipher; we called this phase as Dataprepare, controlled by thread Q-man to prepare multiple blocks of data. The second phase is to send bigger chunks of data for ciphering process, managed by D-man. The last phase is to pull status from H/W to adjust the index of block to process. With these three loosely dependent threads, pipelined processing is enforced as the figure 3, to further accelerate the cipher process.

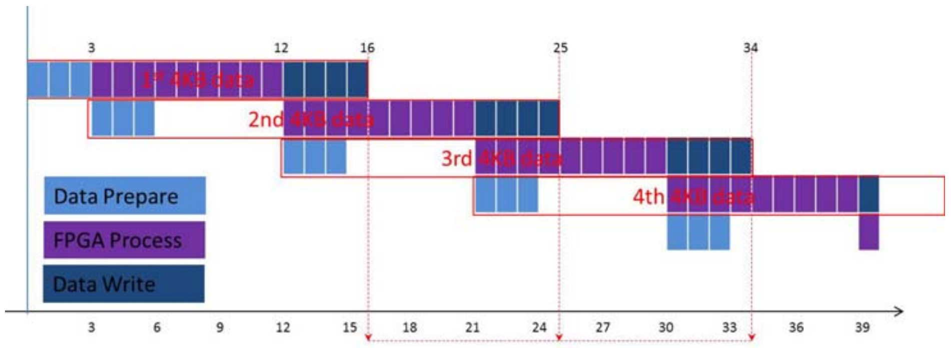

Figure 3. High Throughput Data Encryption with Pipelined Asynchronous Block Cipher Driver

3.5 Accelerating Asynchronous Block Driver

Considering the bottleneck step of DMA read and AES module, two different optimizations are deployed as following: 1) Split DMA read operation into two substage: send DMA read request and receive complete data sending package from CPU; 2) Incorporate two FIFO for pingpang operation on encryption module. AES module takes data from ingress fifo and then do cipher processing for about 20 cycles, two-FIFO pingpang operation can eliminate this bottleneck to 10 cycles. With these two optimizations, pipeline efficiency could be close to 1 with big data protection.

To fully use the feature of pipeline working style of H/W, the device driver is further developed to expand the buffers and let the buffers working as a circular list, where each slots contain index associated with H/W frame slots, and temp pointer point to corresponding request waiting to be processed, as Figure 4.

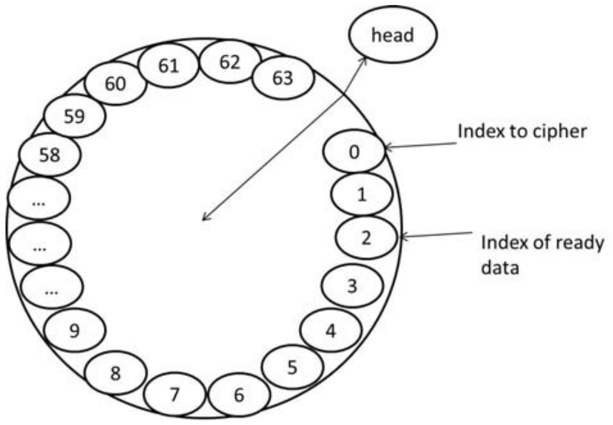

Figure 4. Ring buffer to accelerate cipher

3.6 Cylinder architecture for multiple FPGA

Due to the effect that each request contains all necessary information, it's possible to further create multi-channel DMA to further eliminate the data transfer delay. Based on the ring architecture illustrate above, the driver can further extend to Cylinder architecture version by extend current structure into multi- dimensions array structure, each ring structure corresponding to one FPGA card, with multiple FPGA card or multiple processes on one FPGA card, the performance will be further increase.

4. Experiment Results and Analysis

4.1 Experiment Setup

In our system, two normal computers with 8x PCIe slot is used as an encryption engine and a key manager. On the encryption engine machine, a Xilinx ML605 Demo board with virtex 6 FPGA chip is used for the FPGA-based encryption. We did data protection performance test on 4KB data.

4.2 Comparison among software encryption, device driver with FPGA, optimized driver with FPGA

First, we check the throughput performance without encryption on 4KB data writing of 198MB/s. With this baseline, we did test on data protection throughput on software encryption, basic driver with FPGA version and optimized driver with FPGA. 1) Software version, dm-crypt calls the crypto API for data protection, the maximum throughput on 4KB data with dd tool is 102MB/s; 2) Device driver with FPGA version with 101.39MB/s; 3) Optimized device driver pipelined with FPGA version can support throughput of 367MB/s

Improve the performance by pipelining the different steps. Considering the bottleneck step of DMA read and AES module, two different optimizations are deployed as following: 1) Split DMA read operation into two substage: send DMA read request and receive complete data sending package from CPU; 2) Incorporate two FIFO for pingpang operation on encryption module. AES module takes data from ingress fifo and then do cipher processing for about 20 cycles, two-FIFO pingpang operation can eliminate this bottleneck to 10 cycles. With these two optimizations, pipeline efficiency could be close to 1 with big data protection.

5. Conclusion and Future Works

We create a hardware solution to speed up data protection. Possible future works to improve the performance:

- Setting interrupt scheme with hardware, thus reduce cpu burden of polling check status register.
- Instead of change status number, let hardware directly change CTI pointer (i.e. modify address of CTI), thus the reading of status and updating of CTI can be combined together.
- Instead of using only one FPGA card, upgrade to multiple FPGA card, or one FPGA card with multiple thread concurrently processing.
- With RAID support, concurrently processing on multiple disk, further increase performance.
- Instead of Copy end data from SG list to a linear buffer and send this linear buffer to device, using dmamapsg directly send sg list to device, this will reduce overhead on CPU sufficiently but require necessary hardware support.
- Coming from the idea that dmcrypt may be stacked on top of other device mapper transformations. We may create another general driver layer sitting in between dm-crypt and specific driver.

6. References

[1] Shu Qin Ren, Khin Mi Mi Aung, "PPDS: Privacy Preserved Data Sharing Scheme for Cloud Storage", IJACT, Vol. 4, No. 16, pp. 493 ~ 499, 2012.

[2] Roy Lao Sahagun, Shu Qin Ren, Ho Seng Beng, Khin Mi Mi Aung, "Development of Intelligent Network Storage System with Adaptive Decision-Making", IJACT, Vol. 4, No. 2, pp. 122 ~ 131, 2012.

[3] Yong Khai Leong, Khin Mi Mi Aung, Pantelis Sophoclis Alexopoulos, "Storage System Architecture for Data Centers of the Future", IJACT, Vol. 4, No. 9, pp. 184 ~ 192, 2012

[4] Project dm-crypt http://www.saout.de/misc/dm-crypt/

[5] LUKS for dm-crypt from linux.org http://chakra-linux.org/wiki/index.php/LUKS_for_dm-crypt

[6] Christof Paar, Jan Pelzl , Understanding Cryptography: A Textbook for Students and Practitioners, Springer, 2010

[7] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems", Theory of Cryptography, 6th Theory of Cryptography Conference, TCC (2009): vol.5444 pp.457-473 of Lecture Notes in Computer Science, 2009.

[8] Y. Tang, P. C. Lee, J. S. Lui and R. Perlman, "FADE : Secure Overlay Cloud Storage with File Assured Deletion", Proceeding of the 6th International ICST Conference on Security and Privacy in Communication Networks, SecureComm 2010: pp.380-397, 2010.

[9] Search on Encrypted Data, http://homes.cerias.purdue.edu/~crisn/courses/cs590T/cs590T_lect14_search_encrypted_data.pdf, 2005.

[10] Pairing-Based Cryptography, http://courses.csail.mit.edu/6.897/spring04/L25.pdf, 2004.