

云存储中基于 ORAM 的数据安全访问

李红卫^{1,2}, 古春生^{1,2}, 景征骏^{1,2}, 于志敏^{1,2}

(1 江苏理工学院 计算机工程学院, 江苏 常州 213001;

2 江苏理工学院 云计算与智能信息处理常州市重点实验室, 江苏 常州 213001)

摘 要: 在云存储应用中, 可以利用数据加密技术对存储设备上的数据进行保护, 但加密技术不能隐藏程序对不信任服务器的访问模式, 攻击者可以从访问模式中获取客户的隐私. 提出一种新的茫然 RAM 结构, 利用数据元素位置映射表和存储块位示图实现数据的茫然访问. 将每个数据元素备份在两个文件中, 当一个文件中的数据元素遭到破坏时, 可以从另一个文件中恢复. 以非常少的客户存储容量和 $O(n)$ 服务器存储容量, 使得客户每个数据元素的请求将产生 $O(1)$ 读和写服务器操作.

关键词: 云存储; 茫然 RAM; 访问模式; 校验和

中图分类号: TP309

文献标识码: A

文章编号: 1000-7180(2014)06-0016-05

Access of Data Secure Based on ORAM in Cloud Storage

LI Hong-wei^{1,2}, GU Chun-sheng^{1,2}, JING Zheng-jun^{1,2}, YU Zhi-min^{1,2}

(1 School of Computer Engineering, Jiangsu University of Technology, Changzhou 213001, China;

2 Key Laboratory of Cloud Computing & Intelligent Information Processing of Changzhou City,
Jiangsu University of Technology, Changzhou 213001, China)

Abstract: The data encryption can be employed to protect the data in a storage device, but it cannot be employed to hide the client's data access patterns from an untrusted server in cloud storage applications. The adversary can get the client privacy from access patterns. The paper proposes a new oblivious RAM construction, Using the data element position mapping table and bitmap of storage blocks achieve the oblivious access of data. Each data element has two backups that are stored in two files. When a data element in a file is destroyed, the data element may be recovered from the other file. Each data element request of the client will generate $O(1)$ read and write operations on the server with a small amount of client storage and $O(n)$ server storage.

Key words: cloud storage; oblivious RAM; access pattern; checksum

1 引言

云存储的发展给用户存储数据带来极大的便利, 凡是在有网络的地方用户就能使用和管理自己的数据. 但云存储提供商若管理不善或设备故障就有可能造成数据的丢失或泄露. 例如, 如家汉庭等大批酒店开房记录泄露就说明了数据存储在第三方的

不安全性^[1]. 因此, 云存储的安全会制约云计算的发展^[2]. 传统的加密技术虽然可以隐藏数据的内容, 但客户访问云存储的地址序列可能会包含一些有价值的信息, 攻击者可以从访问地址序列中发现用户的隐私. 因此, 研究数据在第三方存储的安全性以及隐藏数据访问模式的研究具有重要的意义.

收稿日期: 2013-08-23; **修回日期:** 2013-10-11

基金项目: 国家自然科学基金项目(61142007); 常州市云计算与智能信息处理重点实验室项目(CM20123004); 江苏省高校自然科学基金项目(13KJB520005); 江苏省普通高校研究生科研创新计划项目(CXZZ213_0493)

2 相关研究

2.1 基本原理

RAM(Random Access Machine)模型机是由拥有少量寄存器的 CPU 和容量为 n 的存储器构成. CPU 通过 $\text{read}(r)/\text{write}(r, \text{data})$ 指令执行程序实现读/写存储器操作. CPU 访问存储器的地址序列和操作序列称为程序的访问模式^[3].

ORAM(Oblivious RAM)定义:在一个理想的 RAM 模型机上仿真,对于程序的任意两次执行,若输入数据的长度相等,其访问模式是相同的,或在计算上是不可区分的,这样的仿真系统称为安全 ORAM 模型机. 文献[4]最早提出了 ORAM 模型机,其主要目的是隐藏数据的访问模式,即攻击者可以观察到 CPU 访问存储器的地址序列,但无法获知用户的行为. ORAM 模型机同样适用于研究云存储的安全.

隐藏访问模式的主要方法是将客户的数据分割成大小相等的数据元素,与同样大小的若干哑元素混合在一起,按照一定的规律将它们存储在远程存储器上,每当客户访问某个数据元素时,不管是读还是写,ORAM 都会访问一序列的存储块,其中最多有一块是客户需要的数据元素,其他的存储块可能是哑元素,也可能是一个不相关的数据元素. 攻击者无法从客户对远程存储器的访问地址序列上获知客户是读操作还是写操作,以及无法知道哪个存储块是客户实际需要的. 特别地,客户的数据元素在远程存储器上的位置是动态变化的,这只有客户知道,因此在对某个数据元素访问时,每次它所在的存储块位置是不一样的.

安全定义. 设 $y = ((\text{op}_1, u_1, \text{data}_1), (\text{op}_2, u_2, \text{data}_2), \dots, (\text{op}_M, u_M, \text{data}_M))$ 是客户请求访问某一数据块时 ORAM 访问服务器存储块的一个序列,其长度为 M . 其中 op 表示 read 或 write 操作, u 表示服务器虚拟存储地址, data 表示客户存储器地址. 用 $A(y) = (\text{op}_1, \text{op}_2, \dots, \text{op}_M)$ 表示客户访问服务器的操作序列. 如果对于任意两个客户访问序列 y 和 y' ,只要其长度相等,访问服务器的操作序列相等,对任何人(除客户)来说 $A(y)$ 和 $A(y')$ 都是在计算上不可区分的,则称该 ORAM 结构是安全的^[5].

2.2 数据的可恢复性研究

文献[6]首先提出可证明数据拥有协议(Provable Data Possession, PDP),它可以验证外包数据的完整性. 但当文件遭到破坏时,它不能保证文件是

可恢复的. 文献[7]提出可恢复性证明协议(Proof of Retrievability, POR),利用 RS 纠错码^[8]对存储的数据进行编码,当数据遭到破坏时可以利用 RS 纠错码恢复数据. 这两种协议都是对静态数据进行处理,当需要动态更新数据时,就需要对协议进行改进. 另外,如果数据遭到成片破坏,即使使用纠错码也无法恢复. 很多研究者在文献[6-7]的基础上提出了一些改进或新的方案.

2.3 隐藏访问模式的研究

在文献[4]中使用了平方根算法隐藏访问模式,当客户存储量为 $O(1)$,服务器存储量为 $O(n + \sqrt{n})$ 时,均衡负担为 $O(\sqrt{n} \log n)$. 文献[9]提出了金字塔形分层算法,在客户存储量为 $O(1)$,服务器存储量 $O(n \log n)$ 情况下,均衡负担为 $O(\log^3 n)$. 文献[10]是第一个在 ORAM 实用方面进行研究,它是基于金字塔形分层结构,利用加密 Bloom 过滤器和归并排序算法实现 ORAM,当客户存储量为 $O(\sqrt{n})$,服务器存储量 $O(n)$ 时,均衡负担为 $O(\log n \log \log n)$. 但文献[11]认为该模型的实际费用要高得多,因为在 Bloom 过滤器的错误概率非常小的条件下,存储 Bloom 过滤器需要更多的存储空间. 文献[11]提出使用 Cuckoo 散列函数和茫然的希尔排序算法仿真 ORAM,其性能是当客户存储容量为 $O(1)$,服务器容量为 $O(n)$ 时,均衡负担为 $O(\log^2 n)$. 但由于文献[11]在使用 Cuckoo 散列函数时不严谨,致使整个模型存在安全漏洞^[3]. 文献[3]使用客户存储量为 $O(1)$,服务器容量为 $O(n)$ 时,均衡负担为 $O(\log^2 n / \log \log n)$. 文献[3, 10-11]都是在分层结构^[8]的基础上进行的改进,因此,隐藏访问模式都是由访问服务器和洗牌两个阶段构成,其中,洗牌阶段是算法的瓶颈,客户在访问服务器时会因洗牌而阻塞.

文献[12-14]将洗牌过程均摊到每次访存中,优化了最坏情况下的负担. 文献[12]在 ORAM 的实用化方面做了研究,在客户端存储容量为 $O(cn)$ ($0 < c < 1$),服务器存储容量为 $O(n)$ 的情况下,由若干 ORAM 分区构成完整的 ORAM,并采用背景淘汰算法将洗牌操作均摊到每次客户访问操作中,使得客户的一次访问操作仅需访问服务器存储块序列的平均长度在 20 至 35 之间. 它的 ORAM 分区大小是对标准球和盒子的分析得出来的. 假设有 $P = \sqrt{n}$ 个分区,当分区容量为 $\sqrt{n} + o(\sqrt{n})$ 时,有个足够小的溢出概率 $1/\text{poly}(n)$,但这个结论是由下列式子推

导出来的:

如果将 $s = n$ 个球随机扔到 $t = \sqrt{n}$ 个盒子中, 第 i 个盒子中有大于 k 个球的概率是

$$\Pr[\text{第 } i \text{ 个盒子} > k \text{ 个球}] \leq \binom{s}{k} \frac{1}{t^k} \leq \frac{s^k}{k!} \frac{1}{t^k} \leq \left(\frac{\sqrt{n}}{k}\right)^k.$$

显然, 式子的最后一步是错误的. 它的结论因此受到怀疑.

文献[13] 使用 $O(cn)$ ($c < 1$) 客户存储器和 $O(n)$ 服务器存储空间, 隐藏访问模式的负担为 $O(1)$. 文献[14] 提出的 Path ORAM 是第一个得到实用的模型, 它用于设计安全的处理器. 本文在文献[13] 的基础上进一步研究减少客户端存储空间, 并采用双副本的方法保证数据的可恢复性.

在以上所有解决方案中, 都使用了数据加密方案对存储在服务器上的数据加密, 每次将读取的数据元素解密后使用, 当写回不安全的服务器时, 需要加密后写回, 并且采用动态密钥加密数据, 保证相同数据元素加密后结果不一样, 使得攻击者无法基于数据内容进行攻击.

3 ORAM 模型

本文假定 ORAM 模型机是由可信任的客户端和不可信任的服务器构成.

3.1 服务器文件结构

为了保证数据的可恢复性, 采用双备份数据, 即将一个数据块存储在两个不同的位置, 当一个位置上的数据块遭到篡改或损坏时可以从另一个位置恢复. 假设客户的数据由 n 个数据元素构成, 假定一个元素占用一个存储块, 则需要 $2n$ 个服务器存储块存储客户的数据. 另外, 需要在服务器中增加一些哑元素, 将哑元素和客户的数据元素混合在一起, 攻击者无法从表象上区分出哪一个是数据元素哪一个是哑元素, 从而起到隐藏信息的作用. 假设共有 $2n$ 个哑元素. 将 $4n$ 个数据元素和哑元素存放在两个文件中, 假设这两个文件为 f_0 和 f_1 . 每个文件由 n 个数据元素和 n 个哑元素组成, 每个文件都包含完整的客户数据. 将 n 个数据元素均匀随机地分布到 f_0 中的 $2n$ 个元素位置, 剩余的 n 个元素位置存放哑元块. 用同样的方法处理 f_1 . 可以将这两个文件存储在同一个云端, 也可以存放在不同的云端. 存储在不同的云端, 其可靠性更高.

3.2 客户端数据结构

文献[13]用两张数据表映射数据元素在服务器上的位置, 为了进一步减少客户端存储空间, 本文采用位示图的方法描述服务器存储空间的情况. 位示图 $\text{SerBit}[2][2n/32]$ 表示两个文件中各元素的属性, 位示图中的值为 1, 表示该元素位置存储的是数据元素, 否则是哑元素. 数据表 $\text{PosMap}[n]$ 表示每一个数据元素的情况, 它由 4 个域组成, nonce0 和 nonce1 表示新鲜度, 用于数据块的加解密操作; add0 和 add1 分别表示数据元素在两个文件中的位置. 位示图 $\text{ReadBit}[2][2n/32]$ 表示两个文件中各元素的属性, 位示图中的值为 1, 表示该元素已读过, 否则表示该元素写入后还没有读过. 设置缓冲器 tempBuf0 和 tempBuf1 , 用来存放即将写回服务器的数据元素/哑元素, 每个缓冲器最多可以缓存 3 个数据元素. 设置一个缓冲队列 listBuf , 存放客户访问的数据元素, 最多可存放 Q 个数据元素, Q 值越大其安全性就越好.

3.3 访问操作

3.3.1 数据完整性验证

为了提高执行效率, 采用计算数据校验和的方法对数据元素的正确性进行验证. 假设用 B 表示数据元素/哑元素, 用 P 表示校验和, 用 u 表示服务器存储虚地址. 存储块 $M = (B, P)$. 在存储数据元素时先对 M 加密, 然后存储, 可由 nonce0 或 $\text{nonce1} * 3$ 以及 u 生成数据元素密钥. 在存储哑元素时, 先随机生成哑元素, 然后计算哑元素的校验和并由 u 生成密钥加密存储哑元素.

读存储块的操作正好与写存储块的操作相反, 读指定存储块, 若是数据元素, 则用 nonce0 或 $\text{nonce1} * 3$ 以及 u 生成的密钥解密, 验证数据的正确性, 然后使用. 若是哑元素, 则用 u 生成的密钥解密, 并验证数据的正确性. 若数据不正确立即向客户发出警示信号, 并从另一个备份中恢复数据.

3.3.2 数据元素位置映射表 PosMap

在读/写数据元素时, 需要对数据表 PosMap 进行操作. 数据表中的 nonce0 和 nonce1 初值均为 1. 在读数据元素时, 若它们的值相等, 则随机从 f_0 或 f_1 中读取数据块即可, 否则从值小的对应文件中读. 读操作完成后使相应的 nonce0 或 nonce1 的值增 1, 并置 SerBit 相应位为 0, 即将原存储位置设置为哑元素, 置 ReadBit 相应位为 1, 表示该元素最近读过.

若客户是写操作, 假设要写的数据元素为 x , x 在服务器端的备份只有一份, 另一份在客户端的缓

冲队列中(参看 Access() 算法),则将 x 对应的服务器备份设置为哑元素,修改 SerBit 相应位为 0, ReadBit 相应位为 1,并找一个新的哑元块位置,将 x 的一个备份写入该位置,设置 SerBit 相应位为 1, ReadBit 相应位为 0. 在缓冲队列中仍然保留 x ,择机进行第二个备份.

3.4 访问模式

为了隐藏客户的访问模式,对于客户的读或写请求,统一执行 Access(op, u , data)算法实现. 当读数据元素时,执行 Access(read, u , data),所读的数据存放在 data 中;当写元素时,执行 Access(write, u , data),缩写数据元素为 data. Access(op, u , data)算法描述如下:

(1) $p_0 = \text{rand}() \% 2$; // rand()产生随机数, p_0 值为 0 或 1;

(2) $p_1 = \text{rand}() \% 2$;

(3)如果 $p_0 = 0$ 则在 f_0 中随机读取一个数据元素,验证后存入 tempBuf0 缓冲器中;否则在 f_0 中随机读取一个哑元素,验证后丢弃;

(4)如果 $p_1 = 0$ 则在 f_1 中随机读取一个数据元素,验证后存入 tempBuf1 缓冲器中;否则在 f_1 中随机读取一个哑元素,验证后丢弃;

(5)如果 u 已在缓冲器 tempBuf0 或 tempBuf1,或缓冲队列 listBuf 中,则在 f_0 和 f_1 中各读取一个哑元素,验证后丢弃,然后把 u 从缓冲器或缓冲队列中移出,再插入到 listBuf 的队尾;否则在 f_0 或 f_1 中读取指定数据元素,验证后插入到 listBuf 的队尾,同时在 f_1 或 f_0 中读取一个哑元块,验证后丢弃;

(6)如果是写操作,将 data 的内容复制到缓冲队列 listBuf 队尾数据块中,并且将一个备份放入 tempBuf0 或 tempBuf1 缓冲器中,以便更新服务器上的备份;

(7)如果 $p_0 = 1$ 则在 f_0 中随机读取一个实际的数据元素,验证后存入 tempBuf0 缓冲器中;否则在 f_0 中随机读取一个哑元素,验证后丢弃;

(8)如果 $p_1 = 1$ 则在 f_1 中随机读取一个实际的数据元素,验证后存入 tempBuf1 缓冲器中;否则在 f_1 中随机读取一个哑元素,验证后丢弃;

(9)执行 Update()操作,进行写服务器操作;

(10)结束.

随机读数据元素时,要求选择缓冲器和缓冲队列中不存在的数据元素. 在随机读取哑元素时,要求所读的哑元素对应的 ReadBit 位不为 1. 若指定的数据元素 u 已存在于缓冲器或缓冲队列中时,则用

读哑元素代替,以避免攻击者利用时间攻击.

在该算法中随机读取数据元素、哑元素,以及读取指定数据元素的操作混在一起. 攻击者即使了解程序的特性,也无法判定哪一次的读操作是客户需要的. 若数据存储两个云端,可以实现并行访问云存储数据.

前 8 步的执行, tempBuf0 和 tempBuf1 缓冲器中最多存储 2 个数据元素,为了保证下次访问时这两个缓冲器是空的,需要将缓冲器中的数据写回服务器. 每调用一次 Access(),最多有一个数据元素进入 listBuf 缓冲队列,当缓冲队列变满后,需要将一个数据元素写入服务器,以保证下一次调用 Access()时, listBuf 缓冲队列还可以存放一个数据元素. 因此,客户指定访问的数据元素暂时在缓冲队列中存放,不会立即写回服务器,当缓冲队列变满后才将最早进入缓冲队列的数据元素写回服务器,这种淘汰机制是一种茫然的方法. 写服务器的操作由 Update()算法完成,其算法描述如下:

(1)如果 listBuf 队列已满,则将队首数据元素移出队列,根据标识放入 tempBuf0 或 tempBuf1 中;

(2)如果 tempBuf0 和 tempBuf1 缓冲器中数据元素不足 3 块,则用哑元素补足 3 块;

(3)依次为 tempBuf0 和 tempBuf1 缓冲器中的元素分配服务器存储位置;

(4)按所分配的存储块地址进行排序,按排序后的顺序依次写入两个文件中,直到这两个缓冲器为空;

(5)结束.

3.5 性能与安全分析

本文所提 ORAM 模型需要服务器存储空间为 $O(n)$. 客户端所需存储空间由数据结构和缓冲器的大小决定. PosMap 占用的存储空间为 $n(32 + 2 \log_2 n)/8$ 字节,两张位示图需要的存储空间为 $4n/8$ 字节,假设 listBuf 可存放 6 个存储块,则 tempBuf0、tempBuf1 和 listBuf 大约需要 12 存储块. 若 $n = 1\ 024$,每个存储块大小为 32 kB,则客户端所需存储空间大小约为 391 kB,所需服务器存储空间为 128 MB,所用客户空间是服务器存储空间的 0.298%. 访问效率是常量级 $O(1)$,即客户每读/写一个数据块,需要读/写 12 块服务器的存储块,其中 6 次读 6 次写.

根据 ORAM 安全定义可知对服务器的访问操作序列是连续 6 次读操作和连续 6 次写操作. 不论

客户的请求是读还是写,其访问服务器的操作序列都是一样的,因此,ORAM 模型是安全的。

4 结束语

本文提出的常量级 ORAM 模型实现隐藏存储模式,它仅需使用少量的客户端存储空间和 $4n$ 服务器存储容量即可实现茫然访问服务器。采用位示图描述服务器存储块存储属性,使得客户端存储空间的使用量比文献[13]进一步减少。将每一个数据元素备份在两个不同的文件中实现数据的可恢复性,使得数据的可靠性和安全性进一步提高。在已有的文献中,本文性能较优,并可以应用于实践中。但本文客户端所用数据结构与数据元素数量成正比,在面对大数据时,其所需客户端存储空间会因数据量的暴增而变得巨大,因而实现客户端存储空间为常量级仍是今后需要研究的目标之一。在云存储器中实现多方计算并能隐藏访问模式也是今后研究的目标之一。

参考文献:

- [1] WooYun. 漏洞概要[EB/OL]. (2013-10-05). [2013-10-18]. <http://www.wooyun.org/bugs/wooyun-2010-034935>.
- [2] 刘云,曾晋,刘法旺,等. 面向安全风险的云计算测试技术研究[J]. 微电子学与计算机,2013,30(6):15-18.
- [3] Kushilevitz E, Lu S, Ostrovsky R. On the (in)security of hash-based oblivious ram and a new balancing scheme[C]//SODA. Japan;Kyoto,2012.
- [4] Goldreich O. Towards a theory of software protection and simulation by oblivious RAMs [C] //STOC. New York,1987:182-194.
- [5] Shi E, Chan T H H, Stefanov E, et al. Oblivious RAM with $O((\log n)^3)$ worst-case cost[C] // ASIA-CRYPT. Korea;Seoul, 2011: 197-214.
- [6] Ateniese G, Burns R, Curtmolar R, et al. Provable data possession at untrusted stores[C]// Proc of the 14th ACM Conference on Computer and Communications Security. New York: ACM Press, 2007: 598-609.
- [7] Juels A, Kaliski B S. Pors: proofs of retrievability for large files[C] // Proc of ACM-CCS'07. Alexandria, Virginia, USA,2007:584-597.
- [8] 余林琛,章巍,林强,等. RS 纠删码在云存储中的应用[J]. 微电子学与计算机,2011,28(8):234-236.
- [9] Ostrovsky R. Eficient computation on oblivious RAMs [C]//STOC. USA;Baltimore,1990:514-523.
- [10] Williams P, Sion R, Carbunar B. Building castles out of mud: practical access pattern privacy and correctness on untrusted storage[C]// ACM Conference on Computer and Communications Security. USA;Cornell University,2008:139-148.
- [11] Pinkas B, Reinman T. Oblivious ram revisited[C]// CRYPTO. California, USA, 2010:502-519.
- [12] Stefanov E, Shi E, Song D. Towards practical oblivious RAM[C]// NDSS. USA;California,2012.
- [13] 李红卫,古春生,白凤娥. 安全访问外包数据的研究[J]. 电子技术应用,2013, 39(7):54-56.
- [14] Stefanov E, Dijk M, Shi E, et, al. Path o-ram: an extremely simple oblivious ram protocol [EB/OL]. [2013-10-18]. <http://eprint.iacr.org/2013/280.pdf>.

作者简介:



李红卫 男,(1966—),硕士,副教授.研究方向为嵌入式软件与云计算安全。



古春生 男,(1971—),博士,副教授.研究方向为公钥密码。



景征骏 男,(1978—),博士研究生,讲师.研究方向为网络与信息安全、密码学应用。



于志敏 男,(1973—),硕士,讲师.研究方向为网络与信息安全。