ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.10, October 2010, pp.2631−2641
doi: 10.3724/SP.J.1001.2010.03661

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于门限签名方案的 BQS 系统的服务器协议*

荆继武[1], 王 晶[1,2], 林璟锵[1+], 谢永泉[1], 顾 青[3]

[1](中国科学院 研究生院 信息安全国家重点实验室,北京 100049)

[2](中国科学技术大学 电子工程与信息科学系,安徽 合肥 230027)

[3](国家 863 计划信息安全基础设施研究中心,上海 200336)

Server Protocols of Byzantine Quorum Systems Implemented Utilizing Threshold Signature Schemes

JING Ji-Wu[1], WANG Jing[1,2], LIN Jing-Qiang[1+], XIE Yong-Quan[1], GU Qing[3]

[1](State Key Laboratory of Information Security, Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

[2](Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China)

[3](Information Security Infrastructure Research Center of National 863 Program, Shanghai 200336, China)

+ Corresponding author: E-mail: linjq@lois.cn

Jing JW, Wang J, Lin JQ, Xie YQ, Gu Q. Server protocols of Byzantine quorum systems implemented utilizing threshold signature schemes. *Journal of Software*, 2010,21(10):2631−2641. http://www.jos.org.cn/1000-9825/3661.htm

Abstract: Byzantine quorum systems (BQSs) provide attack-resilient storage services over asynchronous channels and tolerate *f* servers' Byzantine failures. COCA (Cornell online certification authority) and CODEX (COrnell Data EXchange) have designed a server protocol to implement BQSs threshold signature schemes (TSS-BQSs), which can execute proactive recovery, conveniently, and can simplify the key management and communication of clients. Under the same system model and channel assumptions, a new server protocol that satisfies the security requirements of TSS-BQS is proposed. The proposed protocol involves less rounds of communication and performs better than that of COCA/CODEX in the case of concurrent read-write operations.

Key words: Byzantine quorum system; threshold signature scheme; attack-resilience; server protocol; fault-tolerance

摘 要: 利用冗余复制技术,BQS(Byzantine quorum system)系统在异步信道上提供了能容忍 *f* 台服务器拜占庭失效的存储服务.COCA 系统和 CODEX 系统设计了一种结合门限签名方案和 BQS 系统的服务器协议,完成了 TSS-BQS(threshold signature schemes-BQS)系统.与普通 BQS 系统相比,具有更易于支持 Proactive Recovery,简化客户端密钥管理和客户端通信的优点.基于相同的系统模型和信道假设,提出了一种新的服务器协议,满足 TSS-BQS 系统的安全要求;而且与已有协议相比,该协议只需更少的通信轮数,在读/写并发情况下执行效果更优.

* Supported by the National Natural Science Foundation of China under Grant No.70890084/G021102 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2006AA01Z454 (国家高技术研究发展计划(863))

Received 2009-01-15; Revised 2009-04-29; Accepted 2009-06-01

关键词: 拜占庭选举系统;门限签名方案;攻击容忍;服务器协议;容错
中图法分类号: TP393 文献标识码: A

BQS(Byzantine quorum system)系统[1]是利用冗余复制技术、容忍服务器拜占庭失效(Byzantine failure,即任意失效 arbitrary failure)、运行于异步信道的存储系统.BQS 系统由 n 台服务器组成,客户端每次操作都要与其中任意 $q(1<q<n)$台服务器通信:写操作时,将数据写入 q 台服务器;读操作时,从任意 q 台服务器读出数据,再从中选出正确的结果.BQS 系统的基本原理是:使执行读/写操作的服务器有足够大的交集,从而排除失效服务器的错误影响,保证读出的结果是最后一次写入的数据.更具体地,对于自验证数据(self-verifying data,例如数字证书),为了容忍 f 台失效服务器系统,共需要 $3f+1$ 台服务器,每一次读/写操作在 $2f+1$ 台服务器上进行.

在实现 BQS 系统时,客户端需要与每台服务器之间建立双向安全信道(reliable and authenticated channel)[1].最基本的方式是使用公钥算法来实现该信道[2]:所有客户端和服务器都具有自己的密钥对,发送者对消息进行数字签名,周期性地重发消息、直至收到接收者的确认消息(也带有接收者的数字签名).

COCA(Cornell online certification authority)系统[3]和 CODEX(Cornell data exchange)系统[4]设计了新的通信协议,完成了基于门限签名方案[5−9]的 BQS 系统(本文称为 TSS-BQS(threshold signature schemes-BQS)系统).在 TSS-BQS 系统中,系统配置有一对公私密钥对(称为 Service Key),其私钥由 n 台服务器分享,任意 $h(h<n)$台服务器可使用私钥执行数字签名.对于 TSS-BQS 系统的每一次读/写操作,客户端只需收到具有 Service Key 数字签名的响应消息即完成操作:由 Service Key 门限签名保证该操作已在 q 台服务器上执行且返回正确结果.需要说明的是,TSS-BQS 系统与特定算法无关,可使用多种不同(n,h)门限签名算法.

TSS-BQS 系统也同样运行于异步信道,可容忍 f 台服务器的拜占庭失效,且具有如下优点:

- 客户端配置唯一固定的 Service Key 公钥、不需要配置 n 台服务器的公钥,简化客户端的密钥管理;
- 客户端只需与任意 $f+1$ 台服务器通信,收到 1 台服务器的响应消息后即可结束操作,简化客户端通信;
- 可在不影响客户端的情况下更方便地执行 Proactive Recovery[10]、更新服务器.相比而言,如果普通 BQS 系统更新服务器,就必须同时更新所有客户端上配置的服务器公钥.

本文提出了一种新的服务器协议,具有 TSS-BQS 系统的优点,而且与 COCA/CODEX 系统的原有协议相比,需要更少的通信轮数、在读/写并发情况下执行效果更好.

本文第 1 节介绍相关研究工作.第 2 节给出 TSS-BQS 系统的系统模型和信道假设.第 3 节描述两种服务器协议.第 4 节给出两种协议在安全性、效率和读/写并发等方面的分析.最后是总结.

1 相关工作

Malkhi 和 Reiter 在 1998 年首先给出了容忍 f 台服务器拜占庭失效的、运行于异步信道的存储系统理论[1],被称为 BQS 系统理论.使用不同的配置(分别称为 Masking Quorum System 和 Dissemination Quorum System),BQS 系统可用于存储自验证数据或普通数据(generic data).然后,他们基于 BQS 系统理论实现了 Phalanx 系统[2].在 Phalanx 系统中,所有客户端都需要配置至少 $2f+1$ 台服务器的公钥,每一次读/写操作要同时与至少 $2f+1$ 台服务器通信,接收 $2f+1$ 个响应消息.在 BQS 系统理论的基础上,研究人员还给出了用于同步信道[11,12]、根据信道延迟优化操作[13]、动态参数配置[14,15]、存储秘密分享数据[16,17]等多种 BQS 系统扩展.

COCA 系统[3]首次利用门限签名方案来实现 TSS-BQS 系统,由多台服务器合作对读/写响应消息进行门限签名,从而保证操作已经按照 BQS 系统理论要求在 $2f+1$ 台服务器上执行.基于 TSS-BQS 系统的基本存储功能,COCA 系统又进一步实现了容错的数字证书(即自验证数据)的签发/查询服务.使用相同的 TSS-BQS 系统服务器协议,CODEX 系统[4]实现了容错的机密性数据存储服务.

2 基于门限签名方案的 BQS 系统

我们使用与 COCA/CODEX 系统完全一致的信道假设和系统模型.TSS-BQS 系统运行于异步 Fair Link 信

道:信道可能丢失消息;但当发送者无限地发送消息时,则接收者可以收到消息.攻击者可以篡改、延迟、截获或删除消息.而且,信道没有最大延迟假设(即异步信道).

TSS-BQS系统首先是BQS系统,由n台独立的服务器组成(记为S_i,$n\geq i\geq 1$),存储的数据记为x.每一台服务器S_i都独立地提供读/写操作功能,S_i存储的x值和时戳,记为$[v_i,t_i]$.

根据BQS系统理论,为了容忍f台服务器的拜占庭失效:

- 对自验证数据,系统由$n=3f+1$台服务器构成、客户端每次读/写$q=2f+1$台服务器;
- 对普通数据,系统由$n=4f+1$台服务器构成、客户端每次读/写$q=3f+1$台服务器.受篇幅所限,本文只详细讨论自验证数据的情况.在第5.4节大致说明了普通数据的情况.

注意,由于写操作只在其中的任意q台服务器上执行,且服务器可能处于失效状态,所以各服务器存储的$[v_i,t_i]$可以不一致.对于自验证数据,使用如下*Result*()函数获得正确结果[1]:对读出的$2f+1$个结果,先排除验证无效的数据,然后取出时戳最大的结果.

在TSS-BQS系统中,使用唯一固定的Service Key来实现对读/写响应消息的数字签名和验证.客户端只接受具有Service Key数字签名的响应消息.Service Key私钥(记为*SK*)由n台服务器分享、各服务器具有部分私钥,其中任意$h(1<h<n)$台服务器可利用门限签名方案执行数字签名.

TSS-BQS系统同时还要求,各服务器具有自己的公私密钥对(不同于Service Key的密钥对),专门用于服务器之间的安全通信[3,4].下文在描述服务器协议时,对于服务器之间的通信消息,不再显式地说明服务器密钥对的数字签名和验证过程.客户端只知道Service Key公钥、不知道(也不需要知道)各服务器的公钥.

每一次读/写操作时,客户端周期性地向任意$f+1$台服务器发送请求消息,直至收到响应消息.与普通BQS系统不同,客户端不需要在接收$2f+1$台服务器的响应消息之后才结束操作,客户端只需收到一个带有Service Key数字签名的响应消息即可结束本次操作:

- 写操作时,客户端周期性地重复发送写-请求消息$[\boldsymbol{W\text{-}Req},x,v_w,t_w,p,cred]$,直至收到写-响应消息$[\boldsymbol{W\text{-}Resp},x,v_w,t_w,p]_{SK}$.其中,$[v_w,t_w]$是待写入的值和时戳,$p$是客户端生成的、防重放攻击的随机数,*cred*是表示客户端具有操作权限、执行本次操作的凭据,*cred*与请求消息内容相关,$[.]_{SK}$表示使用Service Key对消息计算数字签名.每一次操作,客户端都要产生不同的p,并计算相应的*cred*.
- 读操作时,客户端周期性地重复发送读-请求消息$[\boldsymbol{R\text{-}Req},x,p,cred]$,直至收到读-响应消息$[\boldsymbol{R\text{-}Resp},x,v_r,t_r,p]_{SK}$.其中,$[v_r,t_r]$是读出的正确值和时戳.

由于最多有f台服务器失效、客户端向$f+1$台服务器发送请求,所以至少会有一台服务器(称为Delegate,记为S_d)会正确地处理客户端的请求,(与其他服务器共同)执行服务器协议,然后将带有Service Key数字签名的响应消息发送给客户端.所以,服务器协议的功能是:在$2f+1$台服务器(包括S_d)上执行读/写操作,生成响应消息,并联合其他h台服务器(包括S_d)对响应消息进行Service Key门限签名.

需要说明的是,Delegate并不是额外的服务器,每台服务器S_i都同时具有Delegate功能,当正确的服务器收到客户端的请求消息时,就自动转为Delegate,执行服务器协议.对于一次读/写操作,可能同时有多台Delegate.

3 服务器协议

下面分别描述COCA/CODEX系统的服务器协议和我们提出的服务器协议.本文中,将COCA/CODEX系统的原有协议记为SP-I,将我们提出的新协议记为SP-II.

需要说明的是,我们主要讨论存储服务(即BQS系统的基本服务),TSS-BQS系统的服务器协议只包括基本的数据读/写操作.在此基础上的应用功能和应用协议,不是本文讨论的范围.例如,COCA系统的完整协议还包括数据产生(即数字证书签发)操作、CODEX系统的完整协议还包括数据的机密性保护(即数据加解密).在第4.5节,我们讨论了在基本读/写操作基础上的应用功能实现.

3.1 SP-I

在COCA/CODEX系统中,签名门限$h=f+1$,即:任意$f+1$台服务器能够合作完成Service Key门限签名.服务

器使用如下协议:

WRITE

(1) 当 S_d 收到写-请求消息[***W-Req***,x,v_w,t_w,p,$cred$]时,验证 $cred$ 有效后,转发给所有服务器.

(2) 当 S_i 收到由 S_d 转发的写-请求消息时,验证 $cred$ 有效后,返回写-确认消息[***W-Ack***,x,v_w,t_w,i,p].其中,i 是服务器的唯一标识.如果 $v_w>v_i$,则 S_i 同时更新自己存储的值和时戳.

(3) S_d 周期性地重复执行(1),直至收到 $2f+1$ 个不同服务器(包括 S_d)的写-确认消息[***W-Ack***,x,v_w,t_w,i,p].

(4) S_d 生成写-门限签名请求消息[***W-TS-Req***,x,v_w,t_w,p,Σ_w],发送给所有服务器.其中,Σ_w 是(3)收到的 $2f+1$ 个不同服务器的写-确认消息.

(5) 当 S_i 收到由 S_d 发送的写-门限签名请求消息时,进行如下检查(即,检查相应的写操作已在 $2f+1$ 台服务器上执行):

a) Σ_w 是来自 $2f+1$ 台不同服务器的写-确认消息;

b) $2f+1$ 个写-确认消息与写-门限签名请求消息具有相同的[x,v_w,t_w,p](即,针对同一个客户端请求而产生的);

各项检查都成功后,S_i 执行门限签名,返回写-门限签名消息[***W-TS***,x,v_w,t_w,i,PS_i,p]给 S_d.其中,PS_i 是 S_i 利用自己的部分私钥对写-响应消息[***W-Resp***,x,v_w,t_w,p]的部分签名.

(6) S_d 周期性地重复执行(4),直至收到 $f+1$ 个不同服务器(包括 S_d)的写-门限签名消息[***W-TS***,x,v_w,t_w,i,PS_i,p],然后将 $f+1$ 个部分签名 PS_i 合成为完整的 Service Key 数字签名,然后将写-响应消息[***W-Resp***,x,v_w,t_w,p]$_{SK}$ 发送给客户端.

结束协议.

READ

(1) 当 S_d 收到读-请求消息[***R-Req***,x,p,$cred$]时,验证 $cred$ 有效后,转发给所有服务器.

(2) 当 S_i 收到由 S_d 转发的读-请求消息时,验证 $cred$ 有效后,返回读-确认消息[***R-Ack***,x,v_i,t_i,i,p].其中,[v_i,t_i]是 S_i 存储的值和时戳.

(3) S_d 周期性地重复执行(1),直至收到 $2f+1$ 个不同服务器(包括 S_d)的读-确认消息[***R-Ack***,x,v_i,t_i,i,p].

(4) S_d 生成读-门限签名请求消息[***R-TS-Req***,x,v_r,t_r,p,Σ_r],发送给所有服务器.其中,[v_r,t_r]是使用 *Result*()函数对 $2f+1$ 个读-确认消息计算得到的正确结果,Σ_r 是(3)收到的 $2f+1$ 个不同服务器的读-确认消息.

(5) 当 S_i 收到由 S_d 发送的读-门限签名请求消息时,进行如下检查(即,检查相应的读操作已在 $2f+1$ 台服务器上执行且返回结果是正确的):

a) Σ_r 是来自 $2f+1$ 台不同服务器的读-确认消息;

b) $2f+1$ 个读-确认消息与读-门限签名请求消息具有相同的[x,p](即,针对同一个客户端请求而产生的);

c) 读-门限签名请求消息中的[v_r,t_r]等于使用 *Result*()函数对 $2f+1$ 个读-确认消息计算得到的正确结果;

各项检查都成功后,S_i 执行门限签名,返回读-门限签名消息[***R-TS***,x,v_r,t_r,i,PS_i,p]给 S_d.其中,PS_i 是 S_i 利用自己的部分私钥对读-响应消息[***R-Resp***,x,v_r,t_r,p]的部分签名.如果 $v_r>v_i$,则 S_i 同时更新自己存储的值和时戳.

(6) S_d 周期性地重复执行(4),直至收到 $f+1$ 个不同服务器(包括 S_d)的读-门限签名消息[***R-TS***,x,v_r,t_r,i,PS_i,p],然后将 $f+1$ 个部分签名 PS_i 合成为完整的 Service Key 数字签名,然后将读-响应消息[***R-Resp***,x,v_r,t_r,p]$_{SK}$ 发送给客户端.

结束协议.

3.2 SP-II

在我们提出的服务器协议(本文中记为 SP-II)中,签名门限 $h=2f+1$,即:具备任意 $2f+1$ 台服务器才能够合作完成 Service Key 门限签名.

WRITE

(1) 当S_d收到写-请求消息[***W-Req***,x,v_w,t_w,p,*cred*]时,验证*cred*有效后,生成写-门限签名请求消息[***W-TS-Req***,x,v_w,t_w,p,*cred*],并发送给所有服务器.

(2) 当S_i收到由S_d转发的写-门限签名请求消息时,验证*cred*有效后,执行门限签名,返回写-门限签名消息[***W-TS***,x,v_w,t_w,i,PS_i,p].其中,PS_i是S_i利用自己的部分私钥对写-响应消息[***W-Resp***,x,v_w,t_w,p]的部分签名.如果$v_w>v_i$,则S_i同时更新自己存储的值和时戳.

(3) S_d周期性地重复执行(1),直至收到$2f+1$个不同服务器(包括S_d)的写-门限签名消息[***W-TS***,x,v_w,t_w,i,PS_i,p].然后将$2f+1$个部分签名PS_i合成为完整的Service Key数字签名,并将写-响应消息[***W-Resp***,x,v_w,t_w, p]$_{SK}$发送给客户端.

结束协议.

READ

(1) 当S_d收到读-请求消息[***R-Req***,x,p,*cred*]时,验证*cred*有效后,生成读-门限签名请求消息[***R-TS-Req***,x,v_r,t_r,p,*cred*],发送给所有服务器.其中,[v_r,t_r]直接设定为S_d存储的值和时戳(见第4.2.1节的进一步讨论).

(2) 当S_i收到由S_d发送的读-门限签名请求消息时,验证*cred*有效后,进行如下检查:

a) [v_r,t_r]等于自己存储的值和时戳[v_i,t_i];

b) 或者,$t_r>t_i$且v_r是有效的自验证数据(S_i同时更新自己存储的值和时戳);

检查成功后,S_i执行门限签名,返回读-门限签名消息[***R-TS***,x,v_r,t_r,i,PS_i,p].其中,PS_i是S_i利用自己的部分私钥对读-响应消息[***R-Resp***,x,v_r,t_r,p]的部分签名.

如果检查失败,则S_i返回读-门限签名拒绝消息[***R-TS-Reject***,x,v_i,t_i,p].其中,[v_i,t_i]是S_i存储的值和时戳.

(3) S_d周期性地重复执行(1),直至:

a) 收到$2f+1$个不同服务器(包括S_d)的读-门限签名消息[***R-TS***,x,v_r,t_r,i,PS_i,p].

S_d将$2f+1$个部分签名PS_i合成为完整的Service Key数字签名,然后将读-响应消息[***R-Resp***,x,v_r,t_r,p]$_{SK}$发送给客户端.

结束协议.

b) 或者,收到$f+1$个不同服务器的读-门限签名拒绝消息[***R-TS-Reject***,x,v_i,t_i,p].

S_d转到(4)执行.

(4) S_d将读-请求消息[***R-Req***,x,p,*cred*]转发给所有服务器.

(5) 当S_i收到由S_d转发的读-请求消息时,验证*cred*有效后,返回读-确认消息[***R-Ack***,x,v_i,t_i,i,p].其中,[v_i,t_i]是S_i存储的值和时戳.

(6) S_d周期性地重复执行(4),直至收到$2f+1$个不同服务器(包括S_d)的读-确认消息[***R-Ack***,x,v_i,t_i,i,p].

S_d使用*Result*()函数对$2f+1$个读-确认消息计算得到的正确结果[v_r,t_r],然后更新自己的值和时戳,转到(1)继续执行(事实上,由于S_d在(3)(b)已收到$f+1$台服务器的值和时戳,所以只需再收到另外f台服务器的读-确认消息,即可使用*Result*()函数更新自己的值和时戳).

在SP-II步骤(2)的读-门限签名请求消息中,必须包含客户端凭据*cred*供服务器S_i验证,以防止如下攻击:

(1) 在没有收到客户端请求的情况下,失效服务器S_f伪装成Delegate,对所有的可能p值都生成读-门限签名请求消息,要求其他服务器共同进行门限签名(假定S_i不要求验证*cred*,直接进行门限签名).

(2) 对于所有的可能p值,S_f都可合成并缓存相应的读-响应消息[***R-Resp***,x,v_r,t_r,p]$_{SK}$,其中[v_r,t_r]是“当前”有效的值和时戳.

(3) 在[v_r,t_r]失效后(即,在某次写操作之后),S_f就可以根据客户端的读-请求消息中的p值,选择事先已经缓存的相应读-响应消息[***R-Resp***,x,v_r,t_r,p]$_{SK}$发送给客户端.

(4) 客户端将会接受该读-响应消息,但此时[v_r,t_r]已经是旧数据.

注意,在SP-I的读-门限签名请求消息中,没有包含客户端凭据*cred*,但包含了$2f+1$个读-确认消息(即Σ_r).因

为读-确认消息只有 S_i 在步骤(2)验证 *cred* 有效后才会产生,所以,检查Σ_r也就相当于隐含地验证 *cred*,同样可以防止上述攻击.

3.3 协议改进

相比于 SP-I,SP-II 主要在如下几个方面进行了改进:

- 签名门限从 f+1 增加为 2f+1.在 SP-I 中,签名门限是 f+1,可以保证至少有一台正确服务器参加门限签名,该服务器会按照协议要求进行检查,从而保证 BQS 存储服务的完整性:即,读/写操作已在 2f+1 台服务器上执行.在 SP-II 中,签名门限是 2f+1,所以当读/写-响应消息的签名完成时,有 2f+1 台服务器执行了相应的读/写操作,保证了 BQS 服务的完整性(见第 4.1 节安全性分析中的服务完整性进一步说明).
- 简化了服务器进行门限签名之前的检查.因为增加了签名门限,服务完整性不再是由参加门限签名的正确服务器的检查来保证,而是由签名门限来保证,从而可以简化服务器的检查:门限签名之前,服务器 S_i 不需要检查 2f+1 个读/写-确认消息.
- 因为简化了服务器的检查,Delegate 可以直接生成读/写-门限签名请求消息、而不需要先与其他服务器进行 1 轮通信获得 2f+1 个读/写-确认消息(用于门限签名之前的检查),所以 SP-II 可减少通信量.

4 服务器协议分析

下面,我们分别从安全性、效率和读/写并发处理这 3 个方面来分析和比较 SP-I 和 SP-II.

4.1 安全性

TSS-BQS 系统的服务器协议必须满足如下条件:

服务可用性.

在各种失效情况下,对于有效的客户端请求,服务器协议的每一步骤均能顺利执行,最终返回具有 Service Key 数字签名的读/写-响应消息.

服务完整性.

当读/写-响应消息完成门限签名时,保证相应操作已在 2f+1 台服务器上执行.对于写操作,保证将[v_w,t_w]写入 2f+1 台服务器;对于读操作,保证读取当前 2f+1 台服务器的值和时戳,且[v_r,t_r]是正确结果.

对于满足服务可用性和服务完整性的服务器协议,当某 1 台 Delegate 收到客户端请求时,就可以与其他服务器共同执行该协议,返回具有 Service Key 数字签名的读/写-响应消息;同时保证:该读/写操作已在 2f+1 台服务器上执行(从而满足 BQS 系统理论的要求,提供容忍 f 台服务器失效的存储服务).

4.1.1 服务可用性

SP-I 顺利执行的条件是 2f+1 个读/写-确认消息(步骤(3))和 f+1 个读/写-门限签名消息(步骤(6)).

SP-II 顺利执行的条件是 2f+1 个写-门限签名消息(写操作步骤(3)),和:

- 2f+1 个读-门限签名消息(读操作步骤(3),当 Delegate 存储正确的值和时戳);
- 或者,f+1 个读-门限签名拒绝消息(读操作步骤(3))和 2f+1 个读-确认消息(读操作步骤(6),当 Delegate 存储旧的值和时戳).

根据系统假设(系统由 3f+1 台服务器构成、最多 f 台处于失效状态),可以知道系统中至少存在 2f+1 台正确服务器(正确服务器会按照协议要求,生成并返回相应的各种消息),上述条件总是可以满足,SP-I 和 SP-II 能够顺利执行、满足服务可用性.

4.1.2 服务完整性

对于 SP-I,分析如下[3,18]:

首先,响应消息的门限签名需要 f+1 台服务器的合作,其中至少会有 1 台服务器会正确地按照 SP-I 步骤(5)的要求进行检查,保证读/写操作已经在 2f+1 台服务器上执行;

其次,该正确服务器会按照 *Result*()函数检查读-响应消息的返回结果[v_r,t_r];

第三,根据 SP-I 读操作步骤(2),服务器只有在收到有效的客户端请求后,才生成相应的读-确认消息,保证返回结果是在服务器当前存储的值和时戳的基础上生成的.

对于 SP-II,分析如下:

首先,响应消息的门限签名需要 $2f+1$ 台服务器合作,可保证读/写操作已经在 $2f+1$ 台服务器上执行;

其次,根据 SP-II 读操作步骤(2),读-响应消息的返回结果$[v_r,t_r]$等于:任意 $2f+1$ 台服务器中的正确服务器所存储的时戳最大的、有效的自验证数据,与 BQS 系统的自验证数据 *Result*()函数[1]的效果相一致;

第三,根据 SP-II 读操作步骤(2),服务器只有在收到有效的客户端请求后,才与自己存储的值和时戳进行比较,并生成相应的门限签名,保证了返回结果是在服务器当前存储的值和时戳的基础上生成的.

所以,SP-I 和 SP-II 都能满足服务完整性.

4.2 效 率

我们分别从通信量和计算量方面来分析 SP-I 和 SP-II 的执行效率.

4.2.1 通信量

在广域网部署情况下,TSS-BQS 系统的执行效率与通信量密切相关.COCA 系统的实验结果表明[3],在 Internet 部署时,约有 88%的服务器操作时间是用于服务器之间的通信(不计与客户端的通信).

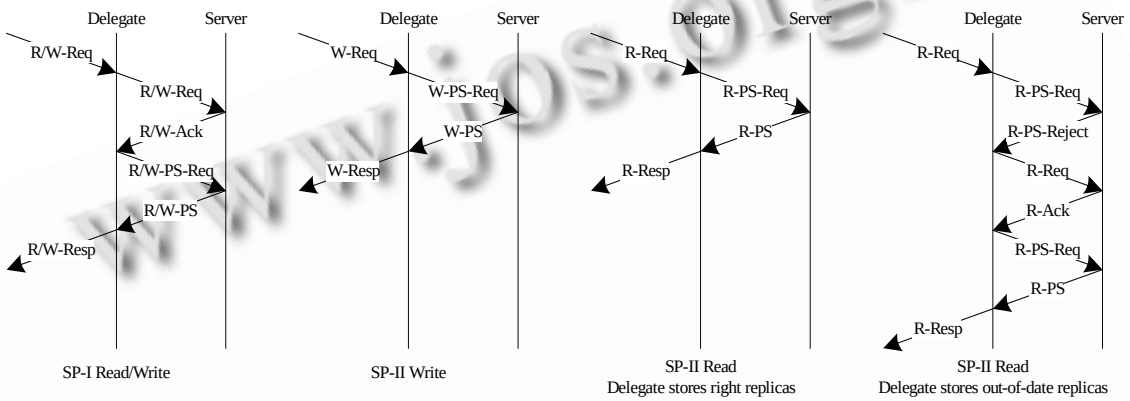

Fig.1 Communication cost of SP-I & SP-II

图 1 服务器协议的通信量

根据上文所述,图 1 给出了 SP-I 和 SP-II 的执行过程.可以看出,SP-I 执行过程(不计与客户端的通信)需要 2 轮通信,SP-II 写操作执行过程只需 1 轮通信.注意:并行的通信过程,只计为 1 轮(如 Delegate 同时向多台服务器发送写-门限签名请求消息).

SP-II 读操作执行过程的通信量则与 Delegate 存储的数据相关:

- 当 Delegate 存储正确数据时,SP-II 读操作执行过程只需 1 轮通信;
- 当 Delegate 存储旧数据时,SP-II 读操作执行过程只需 3 轮通信.

可知,SP-II 写操作的通信量小于 SP-I,但 SP-II 读操作的通信量有可能比 SP-I 要大.

对于 SP-II,由于在写操作时,Delegate 都是使用广播方式发送写-门限签名请求消息,有如下结论:

- 当信道错误率较低时,几乎所有 $3f+1$ 服务器都会收到写-门限签名请求消息并更新数据,则读操作 Delegate(相当于写操作的服务器 S_i)几乎都会存储正确数据,SP-II 读操作执行过程只需 1 轮通信;
- 当信道错误率较高时,几乎只有 $2f+1$ 台服务器收到写-门限签名请求消息并更新数据,此时,读操作 Delegate 很可能存储旧数据,SP-II 读操作执行过程需 3 轮通信.

当信道错误率较高时,我们可以对 SP-II 读操作协议进行如下修正:当 S_d 收到读-请求消息时,就直接假定自

己存储的数据是旧数据,直接从步骤(4)开始执行:先更新数据、然后要求执行门限签名.修正后的 SP-II 读操作协议执行过程如图 2 所示,每次操作只需 2 轮通信(无论 Delegate 存储正确或者旧数据).

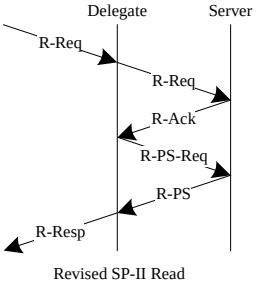

Fig.2 Communication cost of revised SP-II

图 2 修正SP-II的通信量

4.2.2 计算量

COCA 系统的实验结果表明,在局域网部署时,只有约 2%~7%的操作时间用于通信,主要操作时间是 Service Key 门限签名以及服务器通信时的服务器密钥签名和验证.计算量与具体所使用的公钥算法和门限签名算法密切相关.下面,我们以 COCA/CODEX 原型系统使用的门限签名方案[10]为例进行分析.

首先,该方案基于 RSA 算法,RSA 算法的签名验证时间可忽略不计.其次,服务器密钥签名的次数与通信轮数成正比.将各服务器并行执行的服务器签名计算只计为 1 次,则每一次操作中,SP-I 需要 4 次服务器签名,SP-II 写操作需要 2 次服务器签名,SP-II 读操作需要 2~6 次服务器签名.最后,每一次门限签名包括 h 次部分签名计算(模幂计算)和 1 次合成计算(模乘计算,可忽略不计).每一次部分签名计算包括 $(C_n^{h-1}-C_{n-1}^{h-2})$ 次模幂计算(幂指数与私钥长度一致).对于 SP-I 和 SP-II,分别有 $h=f+1$ 和 $h=2f+1$.由 $n=3f+1$ 易验证 $(C_{3f+1}^{f}-C_{3f}^{f-1})=(C_{3f+1}^{2f}-C_{3f}^{2f-1})$.所以,它们的每一次部分签名计算量是相同的.

对于单独的每台服务器而言,SP-II 的每一次门限签名计算与 SP-I 的计算量一样.但是 SP-II 要求 $2f+1$ 服务器参加门限签名计算(SP-I 只要求 $f+1$ 台服务器),所以:

- 在客户端请求很少时,各服务器能够并行地执行部分签名计算,SP-I 和 SP-II 的计算效率大致相同.
- 在客户端请求很多时,各服务器需要调度自己的部分签名计算.由于 SP-II 需要更多的服务器参加计算,所以总体而言,SP-I 的计算效率会优于 SP-II.

假定服务器密钥与 Service Key 的密钥长度一致.由于服务器密钥签名时,可使用中国剩余定理(Chinese remainder theorem,简称 CRT)加速计算,速度是直接模幂计算的 4 倍[19],有如下结果(T_{sign} 表示 1 次 RSA 私钥签名时间,T_{expo} 表示 1 次直接模幂计算时间,$T_{expo}=4\cdot T_{sign}$):

SP-I 计算时间是 $4\cdot T_{sign}+(C_{3f+1}^{f}-C_{3f}^{f-1})\cdot T_{expo}$;

SP-II 计算时间是 $2\sim6\cdot T_{sign}+(C_{3f+1}^{2f}-C_{3f}^{2f-1})\ T_{expo}$.

表 1 列出了当 f 取值较小时 SP-I 和 SP-II 的计算量.

Table 1 Computation cost of SP-I & SP-II

表 1 服务器协议的计算量

f	SP-I write/read	SP-II write	SP-II read
1	$16\cdot T_{sign}$	$14\cdot T_{sign}$	$14\sim18\cdot T_{sign}$
2	$64\cdot T_{sign}$	$62\cdot T_{sign}$	$62\sim66\cdot T_{sign}$
3	$340\cdot T_{sign}$	$338\cdot T_{sign}$	$338\sim342\cdot T_{sign}$

4.3 读/写并发

在 BQS 系统中,每次读/写操作在 $2f+1$ 台服务器上执行,需要持续一定时间,很可能出现读/写并发:在写操作(已有数据记为$[v_0,t_0]$,客户端写入$[v_w,t_w]$,$t_w>t_0$)尚未完成时,同时又有其他客户端读取数据(返回$[v_r,t_r]$).

我们分析了两种服务器协议在读/写并发情况下,读-响应消息返回的数据.从服务质量角度而言,客户端希望尽可能地得到新数据$[v_w,t_w]$,而不是$[v_0,t_0]$.

首先,我们给出如下定义,定义读/写操作的时间窗(开始时间和结束时间):

T_{ws}:当第 1 台执行写操作的正确服务器收到$[v_w,t_w]$时,为写-开始时刻 T_{ws}.此时所存储的数据才真正开始发生变化.虽然在T_{ws}之前可能会有失效服务器收到$[v_w,t_w]$,但不一定对数据产生影响(失效服务器可以不按协议操作,不更新数据).

T_{we}:当第 f+1 台执行写操作的正确服务器收到$[v_w,t_w]$时,为写-结束时刻 T_{we}.此时已完成所需最少写操作.注意:BQS 系统要求在 2f+1 台服务器上执行读/写操作,由于可能存在 f 台失效服务器,相当于只需有 f+1 台正确服务器更新数据即可.

T_{rs}:当 Delegate 收到客户端的读-请求消息时,为读-开始时刻 T_{rs}.

T_{re}:当 Delegate 成功确定了读-响应消息的返回结果$[v_r,t_r]$、开始门限签名时,为读-结束时刻 T_{re}.注意:在 T_{re}之前,Delegate 可能会多次确定读-响应消息的返回结果,但却被其他服务器拒绝.

下面,我们分别针对各种读/写并发情况进行分析.

4.3.1 $T_{ws}<T_{rs}<T_{we}<T_{re}$ 或 $T_{rs}<T_{ws}<T_{we}<T_{re}$

写操作在读操作之前结束($T_{we}<T_{re}$),SP-I 读操作可能返回$[v_w,t_w]$或$[v_0,t_0]$,SP-II 读操作总是返回$[v_w,t_w]$.

对于 SP-I,在读操作开始(T_{rs})时,写操作尚未结束,Delegate 从 2f+1 台服务器收到的读-确认消息中可能只包含$[v_0,t_0]$,不包含$[v_w,t_w]$(例如,可能只有一台正确服务器收到$[v_w,t_w]$、但它不在发送读-确认消息的 2f+1 台服务器之列).参加读操作门限签名的服务器数量是 f+1,其中可能包含 f 台失效服务器和 1 台正确服务器(没有收到$[v_w,t_w]$).那么,门限签名仍可顺利完成,SP-I 读操作返回$[v_0,t_0]$.另一方面,如果 Delegate 从 2f+1 台服务器收到的读-确认消息中包含$[v_w,t_w]$,则 SP-I 读操作返回$[v_w,t_w]$.

对于 SP-II,执行读操作门限签名的 2f+1 台服务器会检查返回结果$[v_r,t_r]$,保证$[v_r,t_r]$与自己存储的数据相同或者有更大的时戳.在写操作结束 T_{we} 时,至少已有 f+1 台正确服务器存储了$[v_w,t_w]$.因为系统共有 3f+1 台服务器,签名门限是 2f+1,至少会有一台存储$[v_w,t_w]$的正确服务器参加读操作门限签名(即,也检查返回结果$[v_r,t_r]$).所以有 $t_r\geqslant t_w$,SP-II 总是返回$[v_w,t_w]$.

4.3.2 $T_{ws}<T_{rs}<T_{re}<T_{we}$ 或 $T_{rs}<T_{ws}<T_{re}<T_{we}$

若写操作在读操作之后结束($T_{re}<T_{we}$),则 SP-I 读操作和 SP-II 读操作都可能返回$[v_w,t_w]$或$[v_0,t_0]$.虽然在读操作结束(T_{re})时,已经有正确服务器收到$[v_w,t_w]$,但是该正确服务器可能不参加读操作的任何步骤(例如,只有一台正确服务器更新$[v_w,t_w]$,但它不在读操作的 2f+1 台服务器之列,也没有参加门限签名),所以 SP-I 读操作和 SP-II 读操作都可能返回$[v_w,t_w]$或$[v_0,t_0]$.

4.4 普通数据支持

在前文描述中,我们假定 TSS-BQS 系统存储自验证数据.事实上,对于 SP-I 和 SP-II 只需进行简单改进,也可以支持普通数据(系统由 4f+1 台服务器组成,每次读/写操作在 3f+1 台服务器上执行,*Result*()函数也要换为相应的函数[1]):

对于 SP-I,签名门限 h 仍是 f+1,只是 Delegate 每次处理的读/写-确认消息数量增加为 3f+1.

对于 SP-II,签名门限 h 从 2f+1 增加为 3f+1.而且在读操作步骤(2),当 $t_r>t_i$ 时,服务器 S_i 无法直接判断$[v_r,t_r]$的正确性,需要与 3f+1 服务器通信,才能检查$[v_r,t_r]$是否正确.此时,SP-II 通信量会大为增加.

4.5 应用功能支持

在 SP-I 存储功能的基础上,COCA 系统和 CODEX 系统分别实现了容错的数字证书签发/查询服务和机密性数据存储服务.在 SP-II 的基础上,同样也可以实现.

首先,在COCA系统的完整协议中,先产生数据(即数字证书签发),然后存储数据,这是先后进行的两个过程;其查询过程完全等同于 TSS-BQS 系统读操作.所以,存储过程的服务器协议可以直接更换为 SP-II.另外,由于数

字证书也使用 Service Key 签发,所以当换为 SP-II 时,数字证书的签发门限也调整为 $2f+1$,不降低其安全性.

其次,在 CODEX 系统中,数据是客户端先加密后写入的,与一般的数据写操作没有差别,所以写操作可以直接更换为 SP-II;读操作则是先由 Delegate 读出数据并组织门限解密得到返回给客户端的结果,然后再要求各服务器对响应消息进行门限签名,相当于 SP-II 中 Delegate 不存储正确数据,要与其他服务器通信读取后再对响应消息进行门限签名的情况.

5 结束语

本文总结了 COCA/CODEX 系统完成的、容忍 f 台服务器失效的 TSS-BQS 系统模型和服务器协议,提出了一种新的服务器协议,并对比分析了两种协议的安全性和效率.分析结果表明,新的服务器协议能够满足 TSS-BQS 系统的安全性要求,且只需更少的通信轮数、在读/写并发情况下的执行效果比原有协议更优.

在 BQS 系统的存储服务上,可进一步设计和实现各种容错的复杂应用服务,如分布式文件系统[20]、发布/分发系统和电子投票系统[2]等.同样,我们也将进一步研究使用 TSS-BQS 系统技术来提供类似的应用服务,并且在 Proactive Recovery,客户端密钥管理和客户端通信方面获得更好的性能.

References:

[1] Malkhi D, Reiter M. Byzantine Quorum systems. Distributed Computing, 1998,11(4):203−213. [doi: 10.1007/s004460050050]

[2] Malkhi D, Reiter M. Secure and scalable replication in Phalanx. In: Singhal M, ed. Proc. the 17th IEEE Symp. on Reliable Distributed Systems. IEEE Computer Society, 1998. 51−60.

[3] Zhou L, Schneider F, Renesse R. COCA: A secure on-line certification authority. ACM Trans. on Computer Systems, 2002,20(4):329−368. [doi: 10.1145/571637.571638]

[4] Marsh M, Schneider F. CODEX: A robust and secure secret distribution system. IEEE Trans. on Dependable and Secure Computing, 2004,1(1):34−47. [doi: 10.1109/TDSC.2004.3]

[5] Desmedt Y, Frankel Y. Shared generation of authenticators and signatures. In: Feigenbaum J, ed. Advances in Cryptology—Crypto'91. Springer-Verlag, 1992. 457−469.

[6] Desmedt Y, Frankel Y. Threshold cryptosystems. In: Brassard G, ed. Advances in Cryptology—Crypto'89. Springer-Verlag, 1990. 307−315.

[7] Frankel Y, Yung M. Distributed public key cryptosystems. In: Imai H, Zheng Y, eds. Proc. the 1st Int'l Workshop on Practice and Theory in Public Key Cryptography. Kanagawa: Springer-Verlag, 1998. 1−13.

[8] Rabin T. A simplified approach to threshold and proactive RSA. In: Krawczyk H, ed. Advances in Cryptology—Crypto'98. Springer-Verlag, 1998. 89−104.

[9] Zhou L, Schneider F, Renesse R. APSS: Proactive secret sharing in asynchronous systems. ACM Trans. on Information and System Security, 2005,8(3):259−286. [doi: 10.1145/1085126.1085127]

[10] Canetti R, Herzberg A. Maintaining security in the presence of transient faults. In: Desmedt Y, ed. Advances in Cryptology—Crypto'94. Springer-Verlag, 1994. 424−438.

[11] Bazzi R. Synchronous Byzantine quorum systems. Distributed Computing, 2000,13(1):45−52. [doi: 10.1007/s004460050004]

[12] Martin JP, Alvisi L, Dahlin D. Small Byzantine quorum systems. In: Fabre J-C, Jahanian F, eds. Proc. the 32nd IEEE/IFIP Int'l Conf. on Dependable Systems and Networks. IEEE Computer Society, 2002. 374−383.

[13] Liu HJ, Jin JW, Lin JQ, Du J. Building an intrusion tolerant repository. Journal of the Graduate School of the Chinese Academy of Sciences, 2006,23(1):46−51 (in Chinese with English abstract).

[14] Alvisi L, Malkhi D, Pierce E, Reiter M, Wright R. Dynamic Byzantine Quorum systems. In: Blough D, Kanoun K, eds. Proc. the 30th IEEE/IFIP Int'l Conf. on Dependable Systems and Networks. New York: IEEE Computer Society, 2000. 283−292.

[15] Martin JP, Alvisi L. A framework for dynamic Byzantine storage. In: Lemos R, Gacek C, Romanovsky A, eds. Proc. the 34th IEEE/IFIP Int'l Conf. on Dependable Systems and Networks. Florence: IEEE Computer Society, 2004. 325−334.

[16] Liu G, Zhou JL, Qin LH, Chen XP. A fault detection mechanism in erasure-code Byzantine fault-tolerance quorum. Computer Science, 2007,34(5):75−78 (in Chinese with English abstract).

[17] Zhang W, Ma JF, Wang LM, Guo YB. Threshold Byzantine quorum system and distributed storage. Acta Electronica Sinica, 2008,36(2):314−319 (in Chinese with English abstract).

[18] Schneider F, Zhou L. Implementing trustworthy services using replicated state machines. IEEE Security & Privacy, 2005,3(5):34−43.

[19] Koc C. High-Speed RSA Implementation. Technical Report, R201, RSA Laboratories, 1994. ftp://ftp.rsasecurity.com/pub/pdfs/tr201.pdf

[20] Castro M, Liskov B. Practical Byzantine fault tolerance and proactive recovery. ACM Trans. on Computer Systems, 2002,20(4):398−461. [doi: 10.1145/571637.571640]

附中文参考文献:

[13] 刘海蛟,荆继武,林璟锵,杜皎.一种入侵容忍的资料库.中国科学院研究生院学报,2006,23(1):46−51.

[16] 刘钢,周敬利,秦磊华,陈小平.纠错码拜占庭容错 Quorum 中错误检测机制.计算机科学,2007,34(5):75−78.

[17] 张薇,马建峰,王良民,郭渊博.门限 Byzantine Quorum 系统及其在分布式存储中的应用.电子学报,2008,36(2):314−319.

荆继武(1964－),男,湖南洪江人,博士,教授,博士生导师,CCF 高级会员,主要研究领域为网络与系统安全.

谢永泉(1975－),男,助理研究员,主要研究领域为信息安全.

王晶(1981－),女,博士生,主要研究领域为信息安全.

顾青(1972－),男,博士,副研究员,主要研究领域为信息安全基础设施.

林璟锵(1978－),男,博士,主要研究领域为网络与系统安全.