基于可验证计算的可信云计算研究

王佳慧[1),3)] 刘川意[2),3)] 王国峰[1),3)] 方滨兴[1),3)]

[1)](北京邮电大学计算机学院 北京 100876)

[2)](北京邮电大学软件学院 北京 100876)

[3)](北京邮电大学可信分布式计算与服务教育部重点实验室 北京 100876)

摘 要 云计算的可信性直接决定了其能否被广泛使用和推广.如果能使得云计算用户验证存储在云平台的数据的完整性或者在云平台执行的程序的正确性,将会大大加快云计算的应用.而可验证计算协议可检测出远程服务器返回的程序执行结果是否正确,且不需要将远程服务器所执行的程序再重新执行一遍.因此,近年来,可验证计算协议引起了学术界和工业界的广泛关注,成为实现可信云计算的一种建设性思路.文中在系统梳理和总结可验证计算协议国内外相关研究的基础上,依据可验证计算协议的实施流程对其按照编译处理和证明系统分类.其中,依据可验证计算协议使用的编译器的复杂程度,分为使用简单编译器的可验证计算协议和使用复杂编译器的可验证计算协议;依据证明系统的分类,主要研究基于交互式证明系统的可验证计算协议和基于论证系统的有预处理的可验证计算协议.随后对依据证明系统划分的每一分类,围绕基本定义、典型协议原理及流程、适用应用场景、性能分析等问题,对基于可验证计算的可信云计算进行了综述.最后,总结和展望了待解决的关键性问题和未来的研究方向.上述工作将对可验证计算协议在云计算中的应用起到一定推动作用.

关键词 云计算;可验证计算;云安全;交互式证明系统;论证系统

中图法分类号 TP309 **DOI 号** 10.11897/SP.J.1016.2016.00286

Review of Trusted Cloud Computing Based on Proof-Based Verifiable Computation

WANG Jia-Hui[1),3)] LIU Chuan-Yi[2),3)] WANG Guo-Feng[1),3)] FANG Bin-Xing[1),3)]

[1)](*School of Computer Science, Beijing University of Posts and Telecommunications, Beijing* 100876)

[2)](*School of Software Engineering, Beijing University of Posts and Telecommunications, Beijing* 100876)

[3)](*Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education, Beijing* 100876)

Abstract To a very great extent, trustworthiness is a critical factor for the large-scale popularity of cloud computing. To instill greater confidence in computations and data outsourced to the cloud service providers, the client should verify the correctness of computation results returned by cloud service providers. Verifiable Computation is the very solution that can let the client check the correctness of a remotely executed computation by inspecting a proof by the remote cloud service providers, without reexecuting the computation. Recent works in verifiable computation have received broad attention in both the academic and industrial research, and the pace of progress in verifiable computation has been rapid. A number of projects have reduced the verifiable computation theory to near-practice in the context of implemented system and have mode it become a constructive approach to trusted cloud computing. The goal of this paper is to survey this blossoming area of research, especially in the cloud computation area. This paper first presents

收稿日期:2015-01-05;在线出版日期:2015-11-02.本课题得到国家自然科学基金(61202081)资助.**王佳慧**,女,1983 年生,博士研究生,主要研究方向为云计算与云安全、数据安全与数据保护.E-mail:jiahw520@gmail.com.**刘川意**,男,1982 年生,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为云计算与云安全、数据安全与数据保护.**王国峰**,男,1988 年生,博士研究生,主要研究方向为云安全.**方滨兴**,男,1960 年生,博士,教授,中国工程院院士,主要研究领域为信息与网络安全、内容安全.

the problem that proof-based verifiable computation is solving in a unified framework, together with some of key theory that has developed to solve it, and then describes design principle and basis theory of verifiable computation protocol. Based on systematic analysis and summary of the related works on verifiable computation, this paper presents a category of verifiable computation protocols according to the two dimensions which are main processes including compile processing and proof system. The verifiable computation protocols in this paper use compiler based on Fairplay and Benjamin compiler to compile the protocol, and generate model of computation for proof system. It can be divided into two approaches according to main process of compiling, as: verifiable computation protocol based on simple compiler and verifiable computation protocol based on complex compiler. It can be divided into two approaches according to proof system, as: verifiable computation protocol based on interactive proof system and verifiable computation protocol with preprocessing based on argument system. We do not discuss verifiable computation without preprocessing based on argument system, and the choice of scope is to make this paper manageable because such protocols are based on short PCPs and still impractical. This paper also covers basic definition, principle and process of typical protocols, application scenarios, performance analysis in each of our classification. At the end of this paper, we summarise some of open questions in this area. The biggest issue in this research area is performance. Also the computational model and underlying theory are a critical area of focus. And other research directions involve changing the model and goal of verifiable computation protocol, and privacy requirements of procotol in the context of cloud computing. All our works will play a role in promoting the further research of cloud computing security based on verifiable computation. And we believe that the real application of these techniques to cloud computing will appear in the next few years with the rapid progress in verifiable computation.

Keywords　cloud computing; verifiable computation; cloud computing security; interactive proof system; argument system

1　引　言

云计算[1]作为一种新兴的网络计算商业服务模式,使得用户可以随时在远端的云服务器存储数据和运行程序.但这种新兴的计算模式在给用户带来诸多便利性的同时,也带来了一些新的安全挑战.用户可能担心云计算平台本身的安全性,比如云平台漏洞和错误配置、管理员的恶意行为等等,而这都可能直接导致用户数据的完整性和隐私性受到危害,导致用户应用程序无法正确执行[2-3].这就产生了一个问题:用户如何相信云提供商执行的程序结果是正确的?如何确保存储在远端的数据的完整性和私密性?

检测远程服务器返回的结果是否正确的传统解决方案有以下几种:(1)采用审计的方法,即随机选取服务器执行的一小部分程序进行验证,这就可能发生错误执行的程序没有被服务器验证的情况,所以说这种方法必须假设错误执行的程序的发生频率是很小的;(2)利用可信硬件[4-5]和远程证明[6-7]来保证远程服务器运行的程序是正确的,但是这种方法必须假设云提供商是完全可信的,由于硬件基础设施是在云提供商的控制之下,如果云提供商内部人员恶意控制了可信硬件(如 CPU、TPM),就无法保障云提供商运行的程序的机密性和可验证性.而且还需要假设存在一个可信链,而运行时可信链的建立在可信计算领域依然是一个难题.事实上,在实际的云计算应用场景中这两个假设通常是无法满足的.在云计算场景中,用户无法完全相信云提供商,即使用户出于声誉的考虑相信云提供商本身,也无法相信其内部管理人员;(3)采用冗余计算的方法,用户可以让多个远程服务器把相同的程序执行多次,然后检测他们返回的结果是否一致[8-12].但这在云计算中也是行不通的,云计算中的软硬件平台配置通常是相同的,而这违背了冗余计算中错误必须

是不相关的假设,且远程服务器很容易窜通,合谋返回一个错误的程序执行结果.

而可证明数据持有[13](Provable Data Possession, PDP)方法和可恢复证明[14](Proof of Retrievability, POR)方法可以用来确保存储在远端的数据的完整性,避免云提供商删除和篡改数据.相比 PDP 方法,POR 除了能确保数据的完整性之外,还能确保数据的可恢复性,但是 PDP 和 POR 无法确保在云提供商端执行的程序的正确性.

另一方面,基于复杂性理论的交互式证明系统(Interactive Proof system,IPs)[15-19]和概率可验证证明系统(Probabilistically Checkable Proof system, PCPs)[20-23]以及密码学理论[24-29]构造的可验证计算协议能以很高的正确率检测出远程服务器返回的程序执行结果是否正确并且不需要对远程服务器(云提供商)做任何假设.可验证计算协议致力于设计验证者与证明者之间的协议,协议允许在计算能力上相对较弱的验证者(如云计算中的用户)将其程序发送到一个计算能力强大的,但不可信的证明者(例如云提供商),并要求证明者执行其发送的程序.所设计的协议应确保证明者不但返回程序的执行结果给验证者,并且使得验证者相信这个程序执行结果是正确的.其主要目标是使得服务器在发送程序执行结果的同时提供程序正确执行的证据,而用户验证证据的过程必须要比用户自己执行程序的开销小(当然有时由于资源比如存储的限制,用户根本无法自己执行程序,在这种情况下是指和假设用户有足够的资源执行程序时的开销相比要小).

近年来,由于云计算的兴起和复杂性相关理论以及密码学理论的进一步发展,可验证计算协议的研究重新受到了关注.Ishai 等人[30]阐述了如何使用简单的 PCPs 和密码学承诺来验证通用计算;Goldwasser 等人[18]把交互式证明系统应用于用特定电路表示的计算中;Gentry 等人[31-32]在全同态加密领域的突破性研究使得起源于 GGP 的通用计算的非交互式协议的构造有了理论依据[22,25].这些研究具有重大意义,基于这些理论的改进使得可验证计算协议最终得以适用于某些特殊构造的应用场景,但是仍然无法适用于通用的云计算场景.

然而可验证计算协议领域的工作目前进展很快,相信在不久的未来将可以应用于云计算的实际场景中.目前使用可验证计算的思路和方法来构建可信云计算,解决云计算中所有权和管理权分离所带来的信任问题的工作还很少.所以本文的主要贡献是系统的梳理和总结可验证计算协议领域的工作.出于本文的研究目的,更关注适用于构建可信云计算的可验证计算协议或者对可信云计算的构建提供建设性思路的可验证计算协议;关注性能合理的,适用于通用场景的,和无条件的(即除了相关的密码学协议假设不做任何假设)可验证计算协议;关注只需要一个证明者,并试图将理论付诸实践的协议.不关注基于特定计算的可验证计算协议[32,34-41],不关注基于多个证明者的可验证计算协议[33],不关注应用同态加密的可验证计算协议,因为同态加密目前还无法适用于实际场景.

接下来的章节中,本文首先介绍可验证计算协议的问题描述和设计原则,接着介绍如何生成协议需要的计算模型,然后依据可验证计算协议的协议流程分别对目前的协议做出分类,之后详细说明每个分类的进展情况.最后,总结和展望该领域目前存在的问题和未来的研究方向,以期对其在国内的研究起到一定推动作用.

2 问题描述和协议设计原则

问题描述:验证者 V 把程序 f 和输入变量 $\boldsymbol{x}$ 发送给证明者 P,P 计算 $f(\boldsymbol{x})$,并把 $f(\boldsymbol{x})$赋值给变量 $\boldsymbol{y}$,返回 $\boldsymbol{y}$ 给 V,然后 V 和 P 以如下方式进行交互:

(1)如果 $\boldsymbol{y}=f(\boldsymbol{x})$,那么 P 应该能向 V 证明 $\boldsymbol{y}$ 的正确性,即使得 V 接受 $\boldsymbol{y}$.其中,证明可以通过回答 V 提出的一些问题完成,也可以通过给 V 提供一个证书完成.

(2)如果 $\boldsymbol{y}\neq f(\boldsymbol{x})$,V 能以很高的概率拒绝接受 $\boldsymbol{y}$.

可验证计算协议的设计必须满足 3 个基本原则:(1)协议应该使得验证者的开销比其在本地执行程序 $f(\boldsymbol{x})$的开销要低,但可以允许证明者为达到协议的目标产生合理的开销,因为提供运行程序的正确性保障本身就需要用户付出一定的代价,在云计算实际场景中,表现为云提供商可能会对需要提供程序正确执行证据的用户收取额外的费用;(2)不能假设证明者完全遵守协议,也就是说证明者可能是恶意的,这和云计算中不能假设云提供商是完全可信的实际场景也是十分吻合的;(3)f 应该是通用程序,然而在具体的协议设计中,可能需要对 f 表示的程序做一些假设,从而通过限制可验证计算协议适用的应用程序种类使得协议的性能达到实际应用场景的要求,但是可验证计算协议的设计原

则依然是尽量能表示通用程序.

通常的安全保障工具比如说病毒检测关注的都是不正确的行为的识别和防范，可验证计算协议则有所不同，其不关心证明者可能的不正确行为，比如犯了什么错误，出现了什么故障等等，而只关心其执行程序的结果是否是正确的，却无法推测程序错误执行的原因. 这和云计算中用户对于程序执行的要求也是相符的.

3　协议流程和关键

3.1　可验证计算协议流程

可验证计算协议的流程主要包括编译处理和证明系统，具体流程如图 1 所示. 首先是编译处理阶段，验证者 V 和证明者 P 将高级语言（比如 C 语言）编写的程序转换成一组布尔电路集（根据协议的不同，也可以是其他计算模型比如算术电路集或者约束集等）. 具体方法在第 4 节详述.

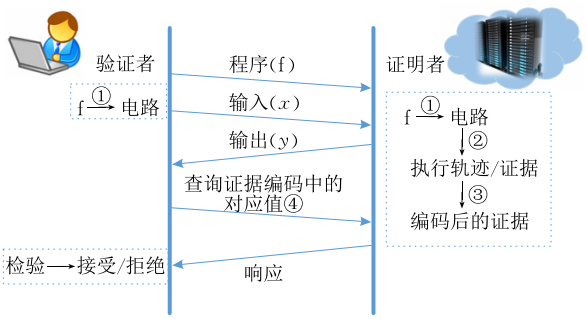

图 1　可验证计算协议流程图

接下来，P 和 V 进行一系列协议交互，不失一般性，这里用布尔电路集 C 表示程序 f. V 把输入变量 x 传输给 P，P 计算 C，输出程序执行结果 y 和 C 正确执行的一组轨迹$\{C,x,y\}$给 V，$\{C,x,y\}$也称为 C 的一个可满足性赋值 z（步骤②）. 其中，C 正确执行的一组轨迹是指 C 的输入线路被赋值为 x，输出线路被赋值为 f(x)时，电路集中所有电路门的赋值集合.

在程序执行的过程中，证明者 P 获得了正确计算电路的执行轨迹$\{C,x,y\}$（步骤②）. 如果 P 声称的输出 y 是不正确的，即 y 不等于 f(x)，那么对于$\{C,x,y\}$，就不可能存在一个有效的执行轨迹（电路 C 正确计算的一个证明）. 因此，如果 P 能够对$\{C,x,y\}$构建一个有效的执行轨迹，那么就一定能使得验证者 V 相信它返回的结果是正确的. 显然，电路正确计算过程中的各个门的赋值本身就能说明存在有效的执行轨迹. 但是，如果需要 V 依次验证所有电路门在计算电路过程中的值，进而确定程序是否正确执行，这个工作量和 V 本地执行 f 是相当的，这就违背了可验证计算协议设计的基本原则.

所以，图中第③步就需要证明者对程序执行轨迹编码，生成一个很长的字符串，并使得不同的执行轨迹生成的编码在所有不同的位置的取值是不相同的. 这样，验证者就可以通过检查随机选择的编码的特定的位置的取值，来验证执行轨迹的有效性，进而对返回的结果采取特定的测试来确定证明者返回的结果是否正确.

但是由于编码很长，读取其需要的时间开销也相应很大，验证者就不可能取回程序执行轨迹的所有编码来完成验证过程. 而且，出于时间开销的考虑，证明者也不可能把执行轨迹的所有编码都写出来，因为验证者只需要查询几个特定位置的值（检查编码的一小部分）就可以完成验证，如果把所有编码都写出来，大部分工作就是被浪费的. 然而，不可能让验证者询问证明者特定位置的编码包含的结果，因为元素的随机性在协议中是非常重要的，可以用来防止证明者的欺骗. 假如验证者的询问事先被证明者了解到，证明者就可能设计好相应的回答来欺骗验证者.

因此，验证者必须仔细慎重的选择（构造）要查询的编码位置（第④步）. 而这也是可验证计算协议设计的核心问题.

3.2　可验证计算协议的理论依据

理解可验证计算协议的原理和流程关键在于理解两个等价关系，如图 2 所示. 如 3.1 节所述，可验证计算协议的流程主要包括编译处理和证明系统. 其中，编译处理阶段，编译器完成高级语言程序到电路集或者约束集（可以看做方程组）等计算模型的转化，其实现的理论依据在于等价关系：程序执行的正确性等价于电路集或者约束集可满足问题.

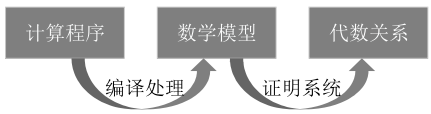

图 2　可验证计算协议的等价关系

定义 1.　电路集是可满足的. 给定输入 x，输出 y，存在一组所有中间电路线的赋值 z，使得电路线所有的电路门都可以同时满足，则称电路集是可满足的，且 z 称为 $C(X=x,Y=y)$的可满足赋值[42].

定义 2.　约束集是可满足的. 给定输入 x，输出 y，存在一组 $Z=z$，使得 $C(X=x, Y=y)$中的约束能

同时满足，则称约束集是可满足的，且 $\boldsymbol{Z}=\boldsymbol{z}$ 称为 $\boldsymbol{C}(\boldsymbol{X}=\boldsymbol{x},\boldsymbol{Y}=\boldsymbol{y})$的可满足赋值[43].

定理 1. 程序执行的正确性等价于约束集/电路集是可满足的，对于所有的 $\boldsymbol{x},\boldsymbol{y},\boldsymbol{y}=\mathrm{f}(\boldsymbol{x})$当且仅当约束集/电路集 $\boldsymbol{C}(\boldsymbol{X}=\boldsymbol{x},\boldsymbol{Y}=\boldsymbol{y})$是可满足的[42-43].

证明系统的主要作用在于依据编译器输出的采用特定计算模型表示的应用程序，设计验证者 V(等同于用户)和证明者 P(等同于云提供商)之间的交互协议，保证用户应用程序执行的正确性. 其理论依据在于使用代数化方法构造如图 2 第 2 个等价关系：电路集或者约束集可满足问题等价于一类很难计算但是容易验证的问题，比如说 QAP(Quadratic Arithmetic Program)理论中的整除关系. 至此，验证程序是否正确执行的问题转化成了验证特定代数关系是否满足的问题.

4 计算模型生成原理流程

为了应用 IPs 和 PCPs 理论构造可验证计算协议，必须先把高级语言程序转换成 IPs 和 PCPs 判定器可以接受的计算模型，比如说电路集和约束集. Cook-Levin 理论[44]表明这种转换理论上是可以的，因为任何程序 f 都可以用图灵机来模拟，同时图灵机可以转换成布尔电路，且不会超过程序的步骤.

目前可验证计算协议中编译器都是基于 Fairplay 和 Benjamin 编译器设计的，常用的计算模型主要有电路集和约束集两种. Fairplay[42]编译器和通常的硬件编译器[45-51]不同，不能使用寄存器，没有时序逻辑，通过 Fairplay Web 网站[52]可以获取该编译器. Fairplay 可以用来把高级语言编写的程序编译成一组布尔电路集，但这种高级语言并不是通常所说的高级语言，而是一种类似 Pascal 或者 C 语言的子集的程序语言，称为(安全函数定义)SFDL 语言. Benjamin 提出的编译器[43]继承并改进了 Fairplay 编译器，用于把高级语言表示的程序编译成一组约束集. 这种编译器也引入了一种类似 SFDL 的高级程序语言，称为扩展函数描述 BFDL.

不失一般性，本文以 Benjamin 编译器为例说明从高级语言程序(C 语言为例)转化成约束集的原理和工作流程.

BFDL 的语法很容易理解，很多地方都是从 C 和 Pascal 语言继承的. BFDL 语言使用 C 风格的语法，是一种静态类型语言，支持类型引用. 其程序结构如图 3：第一部分是类型声明，定义将要使用的数据类型、支持布尔型、整型、结构体和数组.

```
program <program-name>
{
  <type declarations>
  <function declarations>
}
```

图 3 BFDL 程序结构

BFDL 编译器不支持数据下标为变量的程序代码. 图 4 是 Benjamin 编译器的执行流程.

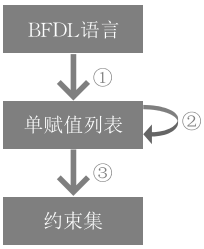

图 4 Benjamin 编译器执行流程

① 使用 BFDL 编译器把 BFDL 语言表示的代码经过函数内联、循环展开等步骤编译成单赋值列表；② 使用修改过的 SFDL 中的优化方法来减少单赋值列表中的赋值描述数量；③ 把赋值描述列表转换为可验证计算协议比如 Zaatar 和 Ginger 可以接受的约束集.

下面以计算一个 $m=14$ 的输入数组中出现目标关键字的次数的程序为例说明从 C 语言到约束集的编译处理过程，图 5 是 C 语言描述示例，图 6 是相应程序依据定义[43]编译的 BFDL 语言描述实例.

```
// compute numbers of key in lists
// constants

const m=14;

int main(int list[m],int key)
{
 int count=0;
 int i;
 for(i=1;i<=m;i++)
 {
  if(list[i]==key)
    count=count+1;
 }
 return count;
}
```

图 5 C 语言程序实例

图 7 和图 8 显示了上述范例从 BFDL 代码到单赋值形式(程序的中间表示)、再从单赋值形式到约束集的转化. 从图 8 中，可以看出图 6 中第 19 行的循环已经展开，if 语句通过构造一个复用转化器表示成单赋值形式. 图 7[43]中列出了布尔表达式和约束的对应关系. 其他诸如条件语句、不等测试、比较操作和约束的对应关系见文献[43]. 而从单赋值形式到约束集的转化由上述规则直接转化. 表 1 给出了布尔表达式和约束的对应关系.

```
/*
 *Compute numbers of key in list
 */
program test {

    // Constants
    const m = 14;

    // Type Definitions
    type Input  =  struct{int<32> key, int<32>[m] list};
    type Output  =  struct{int count};

    // This is the main function
    function Output output(Input X) {

        var int count;
        var int i;
        count=0;
        for(i = 0 to m-1) {
            if(X.list[i] == X.key) {
                count = count + 1;
            }
        }
        output.count = count;
    }

}
```

图 6　BFDL 语言程序实例

表 1　布尔表达式和约束对应表

布尔函数	约束
FALSE	$0-Y_1=0$
NOR	$(1-X_1)\cdot(1-X_2)-Y_1=0$
X_2, NOT X_1	$(1-X_1)\cdot X_2-Y_1=0$
NOT X_1	$1-X_1-Y_1=0$
X_1, NOT X_2	$X_1\cdot(1-X_2)-Y_1=0$
XOR	$(-2X_1)\cdot X_2+X_1+X_2-Y_1=0$
NAND	$(-X_1)\cdot X_2+1-Y_1=0$
AND	$X_1\cdot X_2-Y_1=0$
EQUAL	$(2X_1)\cdot X_2-X_1-X_2+1-Y_1=0$
X_2	$X_2-Y_1=0$
$X_1\rightarrow X_2$	$(-X_1)\cdot(1-X_2)+1-Y_1=0$
OR	$(X_1-1)\cdot(1-X_2)+1-Y_1=0$
TRUE	$1-Y_1=0$

$$
\begin{aligned}
\mathrm{cond}_1 &\leftarrow X.\mathrm{list}[0]!=X.\mathrm{key}\\
\mathrm{count}_1 &\leftarrow \Psi_{\mathrm{MUX}}(\mathrm{cond}_1,0,1)\\
\mathrm{cond}_2 &\leftarrow X.\mathrm{list}[1]!=X.\mathrm{key}\\
\mathrm{count}_2 &\leftarrow \Psi_{\mathrm{MUX}}(\mathrm{cond}_2,\mathrm{count}_1,\mathrm{count}_1+1)\\
&\cdots\\
\mathrm{cond}_{14} &\leftarrow X.\mathrm{list}[13]!=X.\mathrm{key}\\
\mathrm{count}_{14} &\leftarrow \Psi_{\mathrm{MUX}}(\mathrm{cond}_{14},\mathrm{count}_{13},\mathrm{count}_{13}+1)
\end{aligned}
$$

图 7　单赋值形式

$$
\boldsymbol{C}=\left\{\begin{array}{c}
M_1\cdot(X_{\mathrm{list}[0]}-X_{\mathrm{key}})-M_{\mathrm{cond}_1}=0\\
(1-M_{\mathrm{cond}_1})\cdot(X_{\mathrm{list}[0]}-X_{\mathrm{key}})=0\\
M_{\mathrm{cond}_1}\cdot(0-1)+1-M_{\mathrm{count}_1}=0\\
M_2\cdot(X_{\mathrm{list}[1]}-X_{\mathrm{key}})-M_{\mathrm{cond}_2}=0\\
(1-M_{\mathrm{cond}_2})\cdot(X_{\mathrm{list}[1]}-X_{\mathrm{key}})=0\\
M_{\mathrm{cond}_2}\cdot(M_{\mathrm{count}_1}-M_{\mathrm{count}_1}+1)+M_{\mathrm{count}_1}-M_{\mathrm{count}_2}=0\\
\cdots\\
M_{14}\cdot(X_{\mathrm{list}[13]}-X_{\mathrm{key}})-M_{\mathrm{cond}_{14}}=0\\
(1-M_{\mathrm{cond}_{14}})\cdot(X_{\mathrm{list}[13]}-X_{\mathrm{key}})=0\\
M_{\mathrm{cond}_{14}}\cdot(M_{\mathrm{count}_{13}}-M_{\mathrm{count}_{13}}+1)+M_{\mathrm{count}_{13}}-M_{\mathrm{count}_{14}}=0
\end{array}\right\}
$$

图 8　约束集

5　可验证计算协议分类

5.1　依据编译器复杂程度分类

由第 3 节所述，可验证计算协议主要流程包括编译处理和证明系统两个阶段，所以下文将依据协议流程对不同协议分类，如图 9，并说明每种分类的特点和典型协议.

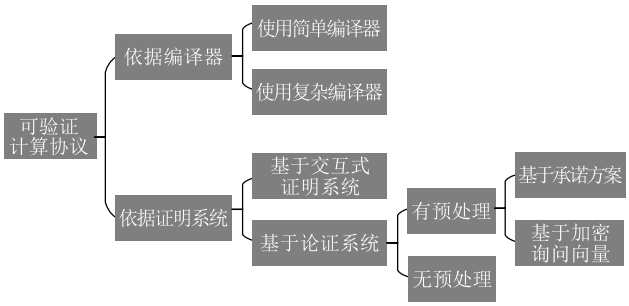

图 9　可验证计算协议分类

本文提到的协议中的编译处理都是直接使用 Fairplay 和 Benjamin 编译器或者对其改进后使用，生成证明系统可以接受的计算模型. 依据可验证计算协议使用的编译器的复杂程度，可以分为简单编译器的可验证计算协议和复杂编译器的可验证计算协议.

简单的编译器是指不支持内存随机存取的编译器，即不考虑内存概念，假设程序的输入都来源于验证者. 简单编译器的可验证计算协议包括 GKR[18]、CMT[53]、Thaler[54]、Allspice[55]、Pepper[56]、Ginger[57]、Zaatar[58] 和 Pinocchio[59] 等，其中 Pinocchio 是第一个直接接受 C 语言程序的协议，而其他协议则需要先将 C 语言程序转化为另一种指定的高级语言比如 BFDL 语言，然后再转化成证明系统可以接受的计算模型.

GKR 使用算术电路作为计算模型，相比较之前协议使用的布尔电路减少了程序编译的开销. CMT、Thaler、Allspice、Pepper 基本沿用了 GKR 中的编译器，且 Pepper 对算术电路进行了简化，Ginger 扩展了算术电路模型表示的程序种类，使得模型包含浮点数类型，不等测试，逻辑表达式，条件语句等等，因而使得模型能表示的程序更加接近于通用程序. Allspice 编译器通过增加了一个静态分析器来自动确定并运行 Zaatar 或 GKR 两个协议中效率较高的一个，增加了协议的可扩展性.

复杂编译器的可验证计算协议包括 Pantry[60] 和 BCGTV[61]. 复杂的编译器支持内存操作，这更符

合实际应用场景.

Pantry 中的编译器改进了 Zaatar 和 Pinocchio 使用的编译器,结合了不可信存储中使用的技术[62-65],使用 Merkle-hash 树来支持内存随机存取.通过构建一个二叉树来表示内存,二叉树的每个叶节点存储相应内存地址的值,每个内部节点存储作用于其子节点的抗冲突哈希函数的值.每当验证者通过(根节点值、内存地址)二元组访问一个内存地址(叶节点)时,证明者可以通过提供沿叶节点到根节点"证明路径"的所有值来"证明"其返回值是正确的.证明者欺骗验证者的唯一方法是通过找到哈希函数中的冲突.由于 Pantry 使用的抗冲突哈希函数的计算函数可以有效地表示成约束集,从而使得内存操作也可以有效的表示成约束集.如果把内存操作也看作普通的程序,就可以实现包含内存操作的程序的可验证计算了.更重要的是,Pantry 支持"远程输入",这使其能更好的支持 MapReduce 程序,且在 MapReduce 程序中为了降低开销定义了 GetBlock和PutBlock 两种元操作来代替构造 Merkle-hash 树.

BCGTV 基于文献[28,58,66],把程序编译成一种特殊的电路表示[67],并把这种计算模型称为 TinyRAM.BCGTV 适用于通用程序,它支持所有的 C 程序,包括独立于数据的循环和 RAM,但不支持"远程输入".然而 BCGTV 的通用性是通过牺牲协议性能来达到的,这使其编译器的开销通常比 Pinocchio 和 Zaatar 编译器开销高好几个数量级.从性能考虑,BCGTV 仍无法用于实际应用程序.

5.2 依据证明系统分类

依据可验证计算协议的证明系统的构造方法不同可将协议分为 3 大类:基于交互式证明系统的可验证计算协议、基于论证系统的有预处理的可验证计算协议和基于论证系统的无预处理的可验证计算协议.本文不讨论基于论证系统的无预处理的可验证计算协议,因为无预处理的论证系统大多基于 short PCPs[68-71],由于其开销过大,仍处于理论研究阶段,还无法用于实际的应用程序.IPs 和 PCPs 理论表明对于给定的数学论断,证明者有能力使得验证者相信论断的正确性.IPs 和 PCPs 理论为可验证计算协议证明系统的构造提供了非常强的保证,使其可以抵御任意的恶意证明者.本文说任意的恶意证明者是指不仅包括故意作恶的证明者,还包括不可预期的行为导致的故障或错误.

交互式证明系统由 Goldwasser、Micali 和 Rackoff[19]以及 Babai 分别提出[15].两种定义是等价的[72],其中 Babai 的系统又称为公开投币系统(Pulic-Coin System).

定义 3. 一个语言 $L\subseteq\{0,1\}^*$ 的交互式证明系统[19]是一个由证明者 P 和验证者 V 组成的交互过程,对于任意的输入串 x,他们的交互过程满足如下的条件:

(1)验证者 V 是一台多项式时间的、带外部信息源的随机图灵机;

(2)证明者 P 的计算能力无限制;P 是一个串函数:$\{0,1\}^*\rightarrow\{0,1\}^*$;

(3)正确性要求:

①完备性.如果 $x\in L$,存在 P,证明者 P 能以至少 2/3 的概率使得验证者 V 接受;

②可靠性.如果 $x\notin L$,则对于任意 P,至多只能以 1/3 的概率使得验证者 V 接受.

文献[19]进一步证明了其完备性可以达到 $1-1/n^k$,可以把交互式证明系统记为 $k-(V,P)(x)$,其中 $|x|=n$,在计算过程中至多交互 $k(n)$次,$k(\cdot)$:$N\rightarrow N$ 是一个多项式时间内可计算函数.为了描述方便,本文把交互式证明系统 $k-(V,P)(x)$称为一个含有外部信息源的判定器,判定器 $V(\cdot,\cdot)$按如下方式工作:

(1)在输入带上放置输入 x,外部信息源记为 π,π 也称为证明串,放置在一条专用证明带上,这里证明带充当证明者 P;

(2)计算过程中产生 w 个地址:$i_1,\cdots,i_w$;依照 $i_1,\cdots,i_w$($i_1,\cdots,i_w$依赖于随机选择),从证明带上获取一个"局部证明":$z=\pi[i_1^r]\cdots[i_w^r]$;

(3)计算 $V(x,z)$.

在多项式时间内验证一个解是否正确的问题存在交互式证明系统[16-17],由上述可见,应用交互式证明系统来设计可验证计算协议,实际上就是设计一个判定器.对于给出的某个程序的计算结果,P 总是要设法让 V 相信程序执行结果的正确性(无论这个结果本身是否正确).交互式证明系统通常含有一个随机操作机制,因为 V 可能有很多问题(即地址)需要 P 来证明,但又没有足够的时间验证,所以只能随机的选择一些问题,同时 V 依靠使用随机串机制来防止 P 的"欺骗".P 通过上述多轮交互过程,判断 P 给出的证明过程是否足以证明程序正确执行.

论证系统[24,34,73-74]实际上也是一种特殊的交互式证明系统,其中,交互的次数最多为两轮,证明者 P 的计算能力限制在多项式时间内.在本文中,我们

将基于论证系统的可验证计算协议和基于交互式证明系统的可验证计算协议分成不同的类别讨论，主要基于以下考虑：(1) 本文中讨论的基于交互式证明系统的可验证计算协议，其交互轮数是指多于两轮的交互，而基于论证系统的可验证计算协议，其交互的轮数被限制在两轮之内，由此带来的协议的性能开销和支持的安全属性都会有很大的不同，在表 2 详细对比；(2) 基于交互式证明系统的可验证计算协议和基于论证系统的可验证计算协议由于协议交互的轮数不同，使得协议具体的构造也有显著的不同，这在本文后续关于协议的具体介绍中会有体现.

表 2　基于交互式证明系统和基于论证系统的协议比较

	交互式证明系统	论证系统
计算能力	无限制	多项式时间
初始阶段开销	无或者开销小	开销大
证明者开销	开销小	开销大
程序种类	并行、规范化	相对通用
交互轮数	多轮	最多两轮
零知识性	不支持	支持
公开可验证性	不支持	支持

文献[30]说明所有有效的论证系统的构造都依赖于 PCPs. PCPs 以及基于此的论证系统实现起来相当复杂，是无法用于实际应用场景的可验证计算协议的，所以基于论证系统的有预处理的可验证计算协议使用的都是特殊构造的 PCPs. 所谓预处理，也称为协议的初始阶段，是需要证明者预先做一个承诺或者验证者预先加密询问向量. 依据预处理的方法不同，可验证计算协议又可以分为使用基于承诺方案的论证系统的可验证计算协议和使用加密询问向量的论证系统的可验证计算协议. 有预处理的协议的优势在于其适用于数据并行度高的应用程序，因为此类情形下，验证者能把初始阶段的开销分摊给同一程序中的多个实例，这样才能使得其开销可以被实际应用程序所接受(即当同一程序被独立应用于许多不同的输入实例时有效)，这和云计算应用场景也十分吻合.

承诺方案[75]是一种基本而用途广泛的密码学原语，是理论密码学中重要的基本模块之一. 承诺方案由两个概率多项式时间(Probabilistic Polynomial-Time，PPT)算法组成，分别称为承诺方和接收方. 承诺方案允许承诺方对选定的值或者声明进行承诺(即计算承诺函数的值). 承诺方案有两个阶段：承诺阶段和公开阶段. 在承诺阶段，承诺方发送其秘密输入(选择的值或者声明)m 的承诺值给接收方；在公开阶段，承诺方可以公开其秘密输入 m. 接收者计算承诺值，并与承诺方发送的承诺值相比较，若相等则承诺被接受否则被拒绝. 承诺方案要满足隐藏性和绑定性两个性质. 隐藏性即接收方根据承诺值不能计算关于承诺方秘密输入 m 的信息. 绑定性即承诺方一旦做出承诺，则无法改变其值，也就是说承诺方不能将承诺值公开为两个不同的 m 和 m'.

基于承诺方案的论证系统的交互一共有两轮，也称为基于交互的论证系统，承诺方的角色由证明者代替，接收方的角色由验证者代替，验证者首先需要证明者对执行轨迹(证据)的编码作出一个承诺. 在第 2 轮中，验证者询问其要查询的证据的位置，然后证明者回答证据在特定位置的值，证明者的回答必须和承诺一致.

使用加密询问向量的论证系统由于验证者在协议的初始阶段发送加密后的询问向量给证明者之后，无需验证者在每个实例再重新发送询问向量，从而也称作无交互的论证系统. 然后，在之后的验证阶段，验证者和证明者使用复杂的密码学协议实现如下的步骤：证明者响应询问向量，但不知道具体要询问的证据的位置，使用证明者的答案，验证者使用 PCP 来验证答案，正如使用基于承诺方案的论证系统的协议一样.

使用基于承诺方案的论证系统的可验证计算协议和使用加密询问向量的论证系统的协议的比较如表 3 所示，前者使用了较少的加密机制，前者验证者发送明确的询问向量，而后者需要证明者计算加密过的询问向量的值. 然而，后者提供了前者没有的功能，支持公开验证性和零知识性，且减少了交互的轮数，允许初始阶段的开销在相同的计算中的所有计算实例分摊.

表 3　基于承诺方案和使用加密询问向量的协议比较

	基于承诺方案	加密询问向量
交互轮数	两轮	无需额外交互
密码学	较少	较多
公开验证	不支持	支持
零知识	不支持	支持
初始阶段	批处理实例分摊	所有实例分摊

基于交互式证明系统的可验证计算协议与基于论证系统的有预处理的协议的比较如表 2 所示. 前者优点主要有 3 个；(1) 即使对有无限计算能力的证明者也是安全的，而后者只是针对多项式时间的证明者是安全的；(2) 交互式证明系统对简单的并行计算能完全不需要预处理阶段，对数据并行的计

算预处理阶段开销也相对较小，而论证系统中验证者固有地需要一个预处理阶段且开销很大；(3) 交互式证明系统在它适用的情况下可使得证明者的开销大大减少，并且事实上对像矩阵乘法这样的基本操作完全可以避免编译成电路的开销. 具体而言，Zaatar 的证明者是比没有正确性验证的计算慢大约 3 个数量级，这与 CMT 中实现的 GKR 中证明者的开销大致相符，但由于生成计算模型的开销，它比 CMT 为“规范化”电路实现的交互式证明系统所能达到的速度要慢且开销大(规范化电路是指易于用加法器和乘法器计算的电路).

交互式证明系统也有其固有的缺点：(1) 它们只适用于深度小的电路(即并行计算)；(2) 它们不支持“非规范化”电路，实际上缺乏对非规范化电路的支持是使用交互式证明系统所固有的. 这意味着，某些种类的计算，如那些涉及许多随机访问存储器或排序/比较操作的计算在交互式证明中是有一定问题的；(3) 交互式证明系统不能支持有些属性像 Pinocchio 的零知识性和公开验证性；(4) 交互式证明系统对证明者和验证者之间的互动需要多轮的对数运算，而论证系统通常只需要一轮或者两轮交互.

下两节将按照可验证计算协议依据证明系统的分类详述每一种分类的相关研究成果，其中，在对每一种分类的典型系统的描述基本遵从如下顺序：从效率最高的但最不通用的协议到更为通用但效率较低的协议.

6 基于交互式证明系统的可验证计算协议

本文首先说明交互式证明系统是如何使验证者 V 确信它接收到的程序执行结果是正确的，如图 10 所示. 假设要执行的程序是计算输入为 $\boldsymbol{x}$ 的函数 f. 首先，验证者在把输入 $\boldsymbol{x}$ 和 f 传输给证明者，同时随机选取关于输入的低阶多项式扩展函数的值[17](比如加权和)作为秘密 s，s 不依赖于要执行的程序，因此无需在输入要执行的程序之前选取秘密 s.

接下来，P 和 V 进行一系列交互(d 轮，d 为电路层数)，这些交互的目的在于 V 控制并引导 P 从生成 $V_0(0,0,\cdots,0)=\boldsymbol{R}_0$ 递归到 $V_d(\boldsymbol{Z}_d)=\boldsymbol{R}_d$(从一层的电路门的值的低阶多项式扩展函数的某个点的值递归到下一层的电路门的值的低阶多项式扩展函数的某个点的值，其中低阶多项式扩展函数是每层的线性组合，如加权和)，V_d 是输入 x 的低阶多项式

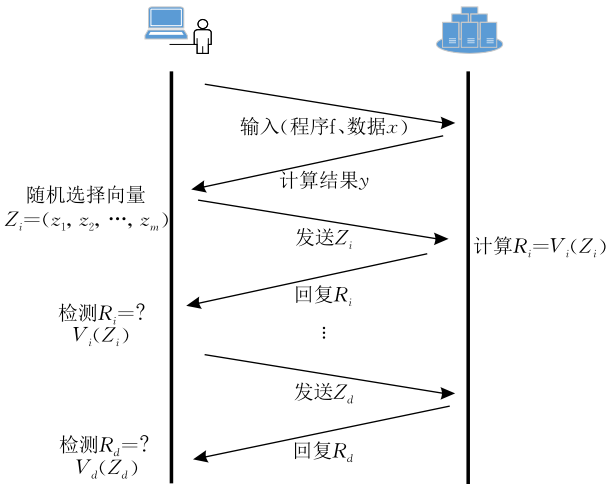

图 10 交互式证明系统流程

扩展函数，V 此时的任务就是计算 V_d 在特定点 Z_d 的函数值，并检测和 P 的回复是否一致. 在这个过程中，V 发送给 P 询问向量 $\boldsymbol{Z}_i=(z_1,z_2,\cdots,z_m)$，P 计算 $\boldsymbol{R}_i=V_i(\boldsymbol{Z}_i)$，用 $\boldsymbol{R}_i$ 回复 V 的询问. 这些询问向量(一共 d 个)都是相关的，V 递归检测 P 对所有询问向量的回复是否一致. V 随机生成的询问向量使得 P 对第一个询问向量的回复包含所执行的程序 f 的结果的声称值. 同样的，P 对最后一个询问的回复包含一个关于 V 的输入变量的低阶多项式扩展函数的某个点的值的声称 $\boldsymbol{R}_d$. 如果 P 对所有向量的回复都是一致的，且声称的值 $\boldsymbol{R}_d$ 和 $\boldsymbol{R}_d$ 的真实值相匹配，然后 P 使得 V 确信其遵守了协议，即正确的执行了程序，因此接受结果. 否则，V 知道 P 在某个点欺骗它，因此拒绝接受.

交互式证明系统在可验证计算协议中的理论应用起源于 GKR 协议[18]，代表协议有 CMT[53]，Allspice[55]，Thaler 等人[54]. Goldwasser、Kalai 和 Rothblum(GKR)在 2008 年提出了一个强大的交互式证明系统用于验证程序执行的正确性. 其中，验证者的成本随着电路的深度呈线性增长(深度小意味着程序的并行性更好)，所以该系统仅实现了在电路具有较小深度时的一个高效验证. 而实际中云计算中的大多计算(应用程序)都表现出大量的并行性.

GKR 协议如图 11 所示，P 和 V 就输入为两个变量(输入变量和声称的输出变量)的函数生成的分层算术电路达成共识. 算术电路就像布尔电路一样，只是其输入是域 F 中的元素，而不是布尔值，且相比之前计算与或非操作，算术电路是执行域 F 上的加乘门运算. 假设电路的输出层是层 d，输入层是层 0. GKR 协议如下迭代运行，每一轮运行对应于电路的每一层. 第 1 轮运行中，V 引导 P 通过一系列的

询问向量和响应从声明电路的输出演绎到声明秘密 s. V 发送的询问向量可以看作有限域 F 中的随机向量，P 的响应是低阶多项式的值，每一个返回的多项式的值依赖于相应的询问向量. 第 1 轮中秘密 s 是层 $d-1$ 的电路门的值的纠错码的几位的组合. 然而 V 并不能直接计算这个 s，要计算 s 必须计算电路的之前所有层，这样会使得可验证计算协议变得毫无意义. 因此，V 必须使得 P 能告诉它 s 的值. 同时 V 必须确认 P 返回的 s 是正确的而不是恶意作假的值. 这正是第 2 轮要实现的内容，V 指引 P 从声明秘密 s 到声明 $d-2$ 层电路门值的纠错码中的几位的组合，作为新的秘密 s'. 如此迭代，直到输入层. 这时，秘密正是输入纠错码的几位的组合，V 就能自己预先计算这个秘密.

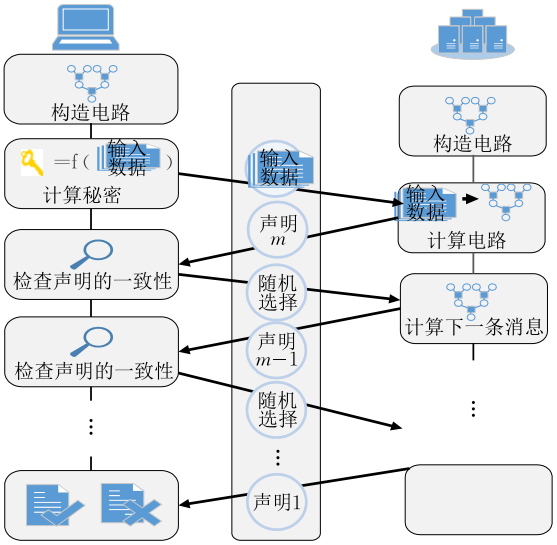

图 11　GKR 协议流程

任何良好的纠错码都满足属性：如果 x 和 x' 在一个地方不同，那么 Enc(x) 和 Enc(x') 在很多地方都不同. 正是由于 Enc(x) 的纠错码具有这样的属性，才使得 V 具有强大且准确的验证能力. 这样，IP 协议中，即使是 P 反转了一个消息中的一位，那么 P 不得不在随后的一个点做出不一致的声称，或者不得不在秘密 s 的值的声称的几乎每个地方都撒谎. 这时 V 就能以很高的频率检测到 P 的欺骗行为.

CMT 协议和 Thaler 协议在 2012 年实现并改进了 GKR 协议.

CMT 协议假设 V_i 为每层电路的评估函数，并定义了两个连接谓词 addi($g1$,$g2$,$g3$) 和 multi($g1$,$g2$,$g3$)，使用连接谓词来表示评估函数，使得评估函数的形式正好与和校验协议一样，从而使用和校验协议的 d 个实例（d 为电路的层数）来校验评估函数. 协议的交互要使得从声明 $V_0(0)=0$ 演绎到声明 V_1，进而演绎到 V_2，直到输入的声明，这时，V 就可以直接检查其是否正确. 协议对 CMT 适用的计算种类性能比较高，很大部分是因为 CMT 协议不需要密码学操作和假设. CMT 使得证明者的运行时间从$O(S\hat{}3)$降低到 $O(S\log S)$，其中 S 为电路门的数目. 实验表明，验证者的成本是非常低的，但 CMT 协议对 GKR 的实现有两个缺点：(1) 证明者的运行时间仍然是一个瓶颈，其相比没有正确性保证的程序多花了大约 3 个数量级的开销；(2) 除非该电路的布线满足“规范化”条件（电路易于加乘门计算），否则验证者需要一个高成本的预处理阶段来“提取”关于电路的布线信息.

Thaler 协议[54]进一步改进了 CMT. Thaler 对于某些种类的程序，证明者的开销可以降低为本地执行程序所需开销的常数级别，这在可验证计算协议领域是非常大的一个改进. 首先，对于“规范化”电路（朴素 naive 矩阵乘法、模式匹配和 FFT 等满足此条件），证明者的运行时间可从 $O(S\log S)$ 降低到 $O(S)$. 具体而言，对于这些电路，证明者现在运行速度比不能保证正确性的电路计算大约慢 10 倍. 第二，Thaler 协议给出了用于矩阵乘法的一个简单协议，完全去除了电路表示的开销，使证明者除执行程序的开销外，只需 $O(n\hat{}2)$ 的开销来证明程序执行结果是正确的. 然而，因为 CMT 协议最初就是为流程序设计的（验证者处理并丢弃输入），使得 Thaler 协议更适用于流程序，且对模型的要求有很多限制，要求电路必须是规范化的相似并行结构. 显然，并不是所有应用程序都可以用这样的模型表示.

2013 年，Allspice 协议[55]对 CMT 的限制条件作了改进，使得其通用性更强，并将 CMT 协议和后边提到的 Zaatar 协议集合成系统的子协议模块，通过一个静态分析器计算各自的效率来决定具体使用哪个协议.

7　基于论证系统的有预处理的可验证计算协议

7.1　使用基于承诺方案的论证系统

基于承诺方案的论证系统，结合承诺方案和线性 PCPs 来构造论证系统，其在可验证计算系统中的应用起源于 IKO 协议[30]，典型协议有 Pepper[56]、Ginger[57] 和 Zaatar[58].

Ishai，Kushilevitz 和 Ostrovsky(IKO) 在 2007 年

提出了一个使用线性 PCPs[20] 代替 short PCPs[68-70] (short PCPs 是之前的论证系统使用的非常复杂的复杂度理论)构造论证系统,从而减少了证据的长度和复杂性,使得可验证计算协议的实用性有了很大的改进. 线性 PCPs 是指证据 π 是线性函数,即 $\pi(\boldsymbol{q}_1+\boldsymbol{q}_2)=\pi(\boldsymbol{q}_1)+\pi(\boldsymbol{q}_2)$,$\boldsymbol{q}_1$ 和 $\boldsymbol{q}_2$ 是询问向量. 一个线性函数 $\pi:\mathbb{F}^n\to\mathbb{F}^b$ 可以看作 $b\times n$ 矩阵 $\boldsymbol{M}$,其中 $\pi(\boldsymbol{q})=\boldsymbol{M}\cdot\boldsymbol{q}$. 特别的,当 $b=1$ 时,π 返回 $\boldsymbol{M}$ 和输入变量的点积. 对于线性函数,求其给定点的值是很容易的.

下面详细描述 IKO 协议原理,主要分为两个部分:假设证据 π 是有限域上的线性函数,第一部分说明 π 的编码和构造线性 PCPs;第二部分说明 V 如何和 π(P)交互.

首先,说明 π 如何编码原始证据(即电路的可满足性赋值)的,V 首先构造一个多项式 P($\boldsymbol{Z}$),使得 P($\boldsymbol{z}$)=0 和高级程序编译生成的电路 C 的可满足赋值 $\boldsymbol{z}$ 是等价的,π 的构造必须能求解此多项式. P($\boldsymbol{Z}$)按如下方式构造:对于 C(一共 s 个电路门)中的每个门,V 构造一个变量 $\boldsymbol{Z}_i\in\{0,1\}$来表示电路门 i 的输出. 同时,V 依照表 4 中的规则构造一个算术约束. 如果门 i 是 $\boldsymbol{Z}_j$ 和 $\boldsymbol{Z}_k$ 的与门,那么 V 构造约束 $\boldsymbol{Z}_i-\boldsymbol{Z}_j\cdot\boldsymbol{Z}_k=0$;如果门 i 是 $\boldsymbol{Z}_j$ 的非门,那么构造约束 $1-(\boldsymbol{Z}_i+\boldsymbol{Z}_j)=0$;如果门 i 是第 j 个输入的输入门,那么构造约束 $\boldsymbol{Z}_i-in_j=0$;针对表示电路输出的最后一个门构造约束 $\boldsymbol{Z}_s-1=0$. 通过组合所有的约束来构造多项式 P($\boldsymbol{Z}$):$\mathrm{P}(\boldsymbol{Z})=\sum_i v_i\cdot Q_i(\boldsymbol{Z})$,其中 $\boldsymbol{Z}=(\boldsymbol{Z}_1,\cdots,\boldsymbol{Z}_s)$,每一个 $Q_i(\boldsymbol{Z})$由一个约束(比如 $\boldsymbol{Z}_i-\boldsymbol{Z}_j\cdot\boldsymbol{Z}_k=0$)给定. v_i 是 V 从有限域 $\mathbb{F}$ 中随机选择的.

表 4 门类型和约束对应表

门(i)类型	约束
与门(j 与 k)	$\boldsymbol{Z}_i-\boldsymbol{Z}_j\cdot\boldsymbol{Z}_k=0$
非门(j 的非门)	$1-(\boldsymbol{Z}_i+\boldsymbol{Z}_j)=0$
输入门(第 j 个)	$\boldsymbol{Z}_i-in_j=0$
输出门	$\boldsymbol{Z}_s-1=0$

因为$\{Q_i(\boldsymbol{Z})\}$都是度为 2 的函数,V 可以把 P($\boldsymbol{Z}$)表示如下:

$$\mathrm{P}(\boldsymbol{Z})=\langle r_2,\boldsymbol{Z}\otimes\boldsymbol{Z}\rangle+\langle r_1,\boldsymbol{Z}\rangle+r_0,$$

式中:$\{r_2,r_1,r_0\}$由$\{Q_i(\boldsymbol{Z})\}$和$\{v_i\}$决定,$r_2\in\mathbb{F}^{s^2}$,$r_1\in\mathbb{F}^s$,$r_0\in\mathbb{F}$. 这样 V 可以通过询问$\langle r_2,\boldsymbol{z}\otimes\boldsymbol{z}\rangle$和$\langle r_1,\boldsymbol{z}\rangle$的值进而计算 P($\boldsymbol{z}$). 为了说明编码,这里令$\langle\boldsymbol{x},\boldsymbol{y}\rangle$表示两个向量 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 的内积,令 $\boldsymbol{x}\otimes\boldsymbol{y}$ 表示外积 $\boldsymbol{x}\cdot\boldsymbol{y}^{\mathrm{T}}$. 由此,可以构造一个正确的证据神谕 π 来编码电路 C 的可满足赋值 $\boldsymbol{z}$,写作$\langle\boldsymbol{z},\boldsymbol{z}\otimes\boldsymbol{z}\rangle$,意味着 $\pi=(\pi^{(1)},\pi^{(2)})$,其中 $\pi^{(1)}(\cdot)=\langle\cdot,\boldsymbol{z}\rangle$,$\pi^{(2)}(\cdot)=\langle\cdot,\boldsymbol{z}\otimes\boldsymbol{z}\rangle$. 以下解释 P($\boldsymbol{z}$)=0 和 $\boldsymbol{z}$ 是可满足赋值的等价性,首先这里假设给 V 的是语义正确但不可满足的 π',也就是说,$\pi'=\langle\boldsymbol{z}',\boldsymbol{z}'\otimes\boldsymbol{z}'\rangle$,但是 $\boldsymbol{z}'$是不可满足赋值. 那么要使得 P($\boldsymbol{z}'$)=0,必然存在至少一个 i' 使得$Q_{i'}(\boldsymbol{z}')\neq0$,也就是说当且仅当 $v_{i'}\cdot Q_{i'}(\boldsymbol{z}')=-\sum_{i\neq i'}v_i\cdot Q_{i'}(\boldsymbol{z}')$,测试才可以通过. 这种概率是很低的,$\{v_i\}$是在 $\boldsymbol{z}'$之后选择的,所以出错的概率不会超过 $1/\mathbb{F}$.

接着说明 V 和 π(P)的交互. 如图 12 所示在 V 发送 f 和 $\boldsymbol{x}$ 并获得 P 执行的结果 $\boldsymbol{y}$ 之后,V 执行承诺方案的承诺阶段,验证者 V 选取随机向量 $\boldsymbol{r}\in\mathbb{F}^s$,应用同态加密算法(这里并不需要假设加密函数是全同态的[76])得到 Enc($\boldsymbol{r}$)发送给证明者 P,P 计算 Enc($\pi(\boldsymbol{r})$)并发送给 V,然后 V 生成针对 PCP 方案的 $\ell=14$ 个询问向量和针对承诺方案的 $\ell=14$ 个询问向量,P 响应 V 的询问向量. 这轮交互也说明了 V 可以通过针对 π 的常数个询问向量就可以验证电路是否可满足.

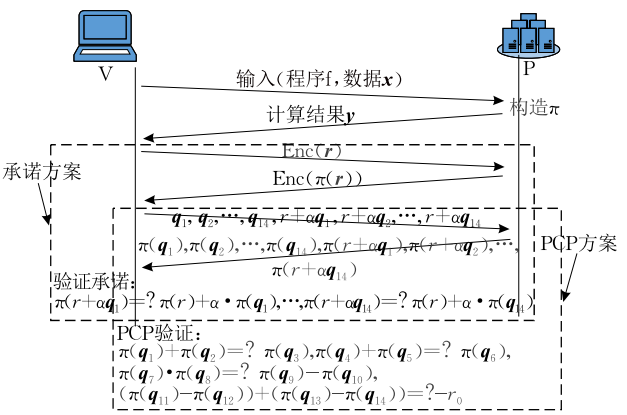

图 12 IKO 协议交互流程

PCP 方案验证分为 4 类:第 1 类称为电路测试,验证 P($\boldsymbol{z}$)=0 是否成立. V 选择 $\boldsymbol{q}_{12}\in_R\mathbb{F}^s$和 $\boldsymbol{q}_{14}\in_R\mathbb{F}^{s^2}$,令 $\boldsymbol{q}_{11}=r_1+\boldsymbol{q}_{12}$,$\boldsymbol{q}_{13}=r_2+\boldsymbol{q}_{14}$,V 验证 $(\pi(\boldsymbol{q}_{11})-\pi(\boldsymbol{q}_{12}))+(\pi(\boldsymbol{q}_{13})-\pi(\boldsymbol{q}_{14}))+r_0$是否等于 0,即 $\pi^{(2)}(r_2)+\pi^{(1)}(r_1)+r_0$是否等于 0. 如果 $\boldsymbol{z}$ 是电路 C 的可满足赋值且 π 被正确的计算了,那么 $\boldsymbol{z}$ 同样满足所有的$\{Q_i(\boldsymbol{Z})\}$,P($\boldsymbol{z}$)=0,验证就可以通过. 如果 $\boldsymbol{z}'$不是可满足赋值,这同时意味着 C 不可满足,那么$\{v_i\}$选择的随机性使得 P($\boldsymbol{z}$)不可能等于 0,V 以极大的概率拒绝. 第 2 类称为线性测试[77-78],验证对于 $\boldsymbol{z}$,$\pi^{(1)}(\cdot)=\langle\cdot,\boldsymbol{z}\rangle$是否成立,对于 u,$\pi^{(2)}(\cdot)=$

$\langle \cdot, u \rangle$是否成立，V 针对 $\pi^{(2)}$ 和 $\pi^{(1)}$ 构造 3 个查询，验证响应结果. $\boldsymbol{q}_1, \boldsymbol{q}_2 \in_R \mathbb{F}^s$ 和 $\boldsymbol{q}_4, \boldsymbol{q}_5 \in_R \mathbb{F}^{s^2}$，令 $\boldsymbol{q}_3 = \boldsymbol{q}_1 + \boldsymbol{q}_2, \boldsymbol{q}_6 = \boldsymbol{q}_4 + \boldsymbol{q}_5$. 如果验证通过，V 就可以确认 $\pi^{(2)}$ 和 $\pi^{(1)}$ 是线性函数. 第 3 类称为二次正交测试，验证两个线性函数的形式是否正确，也就是说检测 $u = z \otimes z$. V 构造 4 个查询并响应验证结果. $\boldsymbol{q}_7, \boldsymbol{q}_8 \in_R \mathbb{F}^s$ 和 $\boldsymbol{q}_{10} \in_R \mathbb{F}^{s^2}$，令 $\boldsymbol{q}_9 = \boldsymbol{q}_7 \otimes \boldsymbol{q}_8 + \boldsymbol{q}_{10}$. 只有这些测试通过，电路测试才有效.

线性测试、二次正交测试和电路测试是为了解除电路测试中 $\pi'^{(2)}$ 和 $\pi'^{(1)}$ 是互相一致的线性函数的假设. 也就是说若假设 π' 是无效的，它编码的一定是不可满足的赋值，而实际上一个恶意构造的神谕 π 可能不满足这个假设.

第 4 类测试称为一致性测试，使用基于同态加密的承诺方案. 测试 P 的响应是否和之前对（π 的）PCPs 的线性特征做出的承诺相一致. 在承诺方案的公开阶段，V 解密得到 $\pi(r)$，选取随机向量 $\boldsymbol{q} \in \mathbb{F}^s$，$\alpha \in \mathbb{F}$ 发送 $(\boldsymbol{q}, r + \alpha \boldsymbol{q})$ 给 P，获得 P 返回的 $\pi(\boldsymbol{q})$ 和 $\pi(r + \alpha \boldsymbol{q})$，$\alpha$ 是从域 $\mathbb{F}$ 中随机选择的. 然后 V 验证 $\pi(r + \alpha \boldsymbol{q}) = \pi(r) + \alpha \pi(\boldsymbol{q})$ 是否成立. 通过并行运行承诺方案的多个实例（每一次 V 需要询问 π 都需要一个实例，仿佛 P 的响应是由若干个非共谋神谕提供的，每一个 PCP 询问由一个神谕提供），IKO 协议把使用线性函数的 PCP 协议转换成论证系统[19,24].

如果 C 是可满足的，那么 V 一定相信 π，否则，C 是不可满足的，V 让测试通过的概率不超过常数 κ（对于任意 π）. 如果重复 ρ 次，构造询问 $\mu = 2\ell \cdot \rho$ 个，那么可靠性错误 $\varepsilon = \kappa^{\rho}$.

然而，IKO 无法用于实际，其对验证者需要高开销的预处理阶段. 即使是对一个 $m \times m$ 的矩阵乘法，验证者的开销也达到 $10^9 \cdot m^6$，要比本地执行矩阵乘法的开销多 $10^9 \cdot m^3$. 这些开销来源于程序的编码编译、大量的密码学承诺和大量的初始阶段开销. 虽然 IKO 协议通过承诺方案降低了证据的复杂性和长度，却使得并行承诺开销很大. 承诺需要密码学操作，IKO 协议过度的调用承诺实例使得协议无法用于实践.

Pepper[56] 和 Ginger[57] 在 2012 年改进 IKO 协议，使得协议有效的支持批处理，大大减小了验证者和证明者的开销. 利用批处理（同一个程序的不同输入实例）来分摊验证者初始阶段的开销. 通常需要大量实例并行才可以使得协议有实际意义.

Pepper 相比 IKO 的直接实现开销降低了 10^{17}. 用算术电路代替布尔电路并简化算术电路减少了程序编码的开销；通过降低承诺实例的调用降低了承诺的开销，如图 13 可以看出，相比 IKO，Pepper 不需要每次询问证据 π 时都调用一次承诺实例，同时提供了更好的安全性；通过批处理来分摊验证者的开销，图 13 中 † 标识的消息在批处理的多个实例中只需执行一次；针对特殊场景降低了 PCP 编码的冗余度从而减少了证明者 P 的开销. Pepper 协议中证明了这些改进并没有改变 IKO 协议的完备性和可靠性. 然而 Pepper 协议的计算开销和网络开销依然很大，适用的应用种类有限，局限于顺序结构的数值计算.

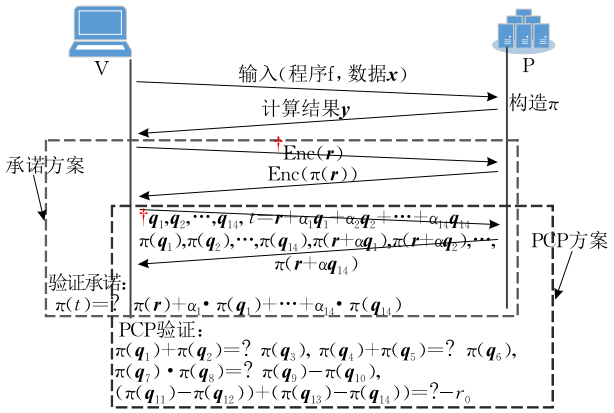

图 13　Pepper 协议交互流程

Ginger 协议[57] 进一步降低了开销，其证明了一致性测试是包含线性测试的，即使是一个恶意的证明者也一定会提供线性的 π，从而去除了图 13 中的 PCP 验证的线性测试. 如图 14 所示，V 以如下方式构造查询：$\boldsymbol{q}_1, \boldsymbol{q}_2 \in_R \mathbb{F}^s \boldsymbol{q}_4 \in_R \mathbb{F}^{s^2}$，令 $\boldsymbol{q}_3 \leftarrow \boldsymbol{q}_1 \otimes \boldsymbol{q}_2 + \boldsymbol{q}_4$，$\boldsymbol{q}_5 \leftarrow r_1 + \boldsymbol{q}_1, \boldsymbol{q}_6 \leftarrow r_2 + \boldsymbol{q}_4$，且在二次正交测试和电路测试中重用了部分询问向量，使得一个实例中 PCP 验证中询问的次数从 13 降到了 6，进一步降低了询问的次数.

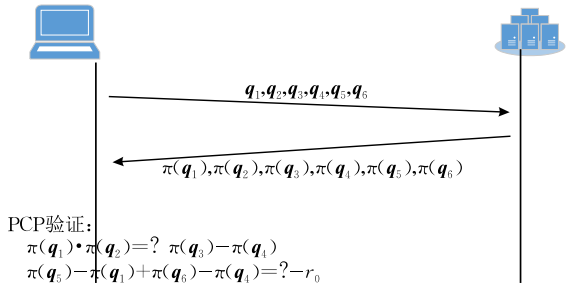

图 14　Ginger 中的 PCP 验证

Ginger 协议扩大了应用的适用类型，使得程序包括浮点数、不等测试、逻辑表达式，条件语句等等. 但仍不能很好的处理循环. 且相比较 Pepper，Ginger编译器可以自动完成绝大部分程序的编译.

然而 Ginger 协议有一个严重的问题,就是证明者的每个实例的工作和验证者生成询问向量的初始阶段的开销是计算规模大小的二次方. Ginger 针对特定程序(比如矩阵乘)通过手工裁剪证据向量和询问向量来解决这个问题,但是这只适用于某些应用场景.

Pepper 和 Ginger 协议仅仅在有重复结构的顺序结构程序是有意义,而且他们针对不同的应用程序需要特殊构造的 PCP. 且为了使得 Pepper 和 Ginger 协议获得好的性能从而能用于实际,在程序的通用性上做了限制,只适用于特定种类的程序. Zaatar 协议保留了 Pepper 和 Ginger 协议的大部分结构,并对其做了改进,引入了(GGPR)[28] 中的基于 QAPs 的线性 PCP 取代了 Arora 等人[20] 中的线性 PCP,和 Ginger 协议的承诺方案相结合,使得验证者的初始阶段的开销和证据 π 的长度等于程序运行时间和程序长度之和的线性函数,从而使得证明者的开销接近线性. 虽然这种编码引入了额外的开销,但这些开销相对 Ginger 协议以及之前的同类协议来说仍旧是最优秀的. 证明者的开销和 Ginger 中最优化的手动构造的程序协议基本一致. 在大部分程序种类中,基本取代了 Pepper 和 Ginger 协议.

Zaatar 协议可以自动将程序编译成约束集,Pepper 和 Ginger 协议都是手动构造的. 基于一些基准测试(排序、聚类分析、最短路径),Zaatar 协议的证明者的性能降低了 3～6 个数量级,验证者的性能也降低了 3～6 个数量级,因而验证者的批处理实例可以降低到上千个计算. 同时由于用户愿意为安全保证付出一定的代价,这使得 Zaatar 协议在实际应用中的开销已经可以接受,且批处理和付费问题等特征和云计算的当前应用也一致. 比如说,大规模的科学模拟计算通常有重复的结构,MapReduce 程序的 map 阶段也如此,但是由于其使用的计算模型的局限性,使得 Zaatar 协议还无法应用于云计算场景,因为它需要验证者本地拥有所有的输入. 为了使得 Zaatar 协议能应用于 MapReduce 程序,需要修改协议,使得协议能够处理远程输入的场景,比如下一节介绍的和不可信存储[64-65,79-80] 集成的 Pantry 协议.

7.2 使用加密询问向量的论证系统

相比使用基于承诺方案的论证系统,使用加密询问向量的论证系统需要验证者预先加密它的询问向量,证明者无法看到询问的实际内容. 随后,在验证阶段,验证者和证明者使用复杂的密码学协议实现如下的流程:证明者回答询问向量,而不能分辨询问向量询问的证据的位置,验证者用密钥解密验证者的响应. 有了证明者的响应,验证者继续执行 PCP 验证,就如使用基于承诺方案的论证系统的协议一样. 这个方法在文献[28,66]中详细说明,文献[59-60]改进和实现了该方法.

Parno 等人[59] 提出 Pinocchio 协议,其第一次使用了加密询问向量的论证系统的方法,这对于用户的隐私有重要的意义. Pinocchio 协议不但实现了 GGPR 中的证据编码,还实现了其复杂的密码学协议,使得 Pinocchio 协议具有零知识性和公开可验证性,任何拥有密钥的用户都可以验证程序是否正确执行,只要其生成的询问向量是正确的. 加密询问向量的论证系统很大的一个优点是可以重用询问向量,大大减少了询问向量生成的时间,而且使得协议中交互的次数更少,在每一个程序的初始阶段之后,验证者只需发送每一个实例的输入数据即可,所以也称为无交互的论证系统.

从定性的分析来看,Pinocchio 协议具有更好的分摊性. 如图 15 所示 Pinocchio 协议可以把每一个程序的初始阶段的开销分摊给未来的所有此程序的实例,而使用基于承诺的论证系统的可验证计算协议只是把每一个程序的初始阶段开销分摊给一次批处理中的所有实例. 虽然复杂的密码学协议的应用相比基于承诺的协议而言给初始阶段带来了一些额外的开销,但这些开销比较小,而且通过分摊可以使开销忽略不计.

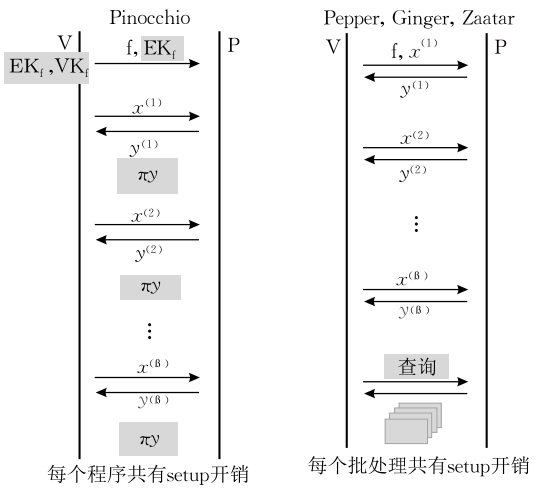

图 15 Pinocchio 和基于承诺方案的协议区别

尽管 Pinocchio 协议已经在可验证计算领域取得了很大的突破,但是其所采用的计算模型仍不能

很好的表示通用程序.首先,协议需要在编译时定义循环的次数.其次,不支持间接内存引用,这就无法表示通用程序中的 RAM 操作.第三,这些程序不支持远程输入,也就是说验证者必须处理所有的输入和输出,这就无法表示 MapReduce 程序,远程数据库查询等等,无法适用于云计算中.

Pantry 协议[60]对其进行了进一步的改进,把 Pinocchio 协议和不可信存储技术[62-65,79-81]相结合,引入了存储的概念,使用 Merkle-hash 树技术来支持内存随机存取,其中 hash 函数采用 Ajtai[82-83],其相比 SHA-1,时间开销只是其 10%.Pantry 协议定义了如图 16 两个基本原语 PutBlock 和 GetBlock,使用编译器将其表示成约束集 C_H 和 $C_{H^{-1}}$,如图 16,然后使用 PutBlock 和 GetBlock 来构建可验证的 MapReduce、可验证的 RAM、远程数据库的可验证的查询.

```
GetBlock(name n)
{
 block <-- read block with name n in block store S;
 assert n == H(block);
 return block;
}

PutBlock(block)
{
 n <-- H(block);
 store (n,block) in block store S;
 return n;
}
```

图 16　Pantry 基本原语

以 MapReduce 为例说明,如图 17 所示,程序员使用 C 语言和 block=GetBlock(name)和 name=PutBlock(block)原语来构建可验证的 MapReduce 框架,只是输入和输出文件用其内容的摘要来表示.如此构造的程序,Pantry 协议则可以直接把程序作为 Pinocchio 协议的输入程序来处理.同时 Pantry 也可以集成 Zaatar 协议,只是如果要提供公开可验证性和零知识性,就必须集成 Pinocchio 协议处理.

Pantry 协议是目前处理验证者不包含所有输入的程序的可验证计算的唯一协议.由于 Pantry 协议中验证者不处理输入,节省了 CPU 和网络开销.但是由于 hash 函数的开销很大,使得 Pantry 协议无法适用于内存密集型程序.而且由于 hash 函数的开销,使得需要很多的计算实例才可以分摊初始阶段的开销.Pantry 对于数据库查询应用,数据库只能有一个写入者,这离实际的数据库应用的可验证计算还有一定的距离.

```
DigestArray Mapper(Digest X)
{
 Block list_in = GetBlock(X);
 Block list_out[NUM_REDUCERS];
 Digest Y[NUM_REDUCERS];

 // invoke programmer-supplied Map
 Map(list_in, &list_out);

 for (i=0; i<NUM_REDUCERS; i++)
     Y[i] = PutBlock(list_out[i]);

 return Y;
}
```

```
Digest Reducer(DigestArray X)
{
 Block list_in[NUM_MAPPERS];
 Block list_out;

 for (i=0; i<NUM_MAPPERS; i++)
     list_in[i] = GetBlock(X[i]);

 // invoke programmer-supplied Reduce
 Reduce(list_in, &list_out);

 Y[i] = PutBlock(list_out[i]);

 return Y;
}
```

图 17　可验证的 MapReduce 框架

8　问题与展望

目前可验证计算协议还只是“玩具”系统,由于性能开销过大,仍无法真正用于通用应用程序和云计算的实际场景中.本文说这些协议接近实际场景,是因为相对于相关理论的直接实现所产生的开销来说,这些协议已经有了质的飞跃.在特定构造的程序中,这些协议还是有意义的.而且,在某些需要牺牲性能来换得安全性的场景下,比如在高确保计算场景中,为了掌握部署在远端的机器的运行是否正常,通常愿意花费比较大的代价.更幸运的是,现有可验证计算协议基于性能的考虑要求证明者有大量空闲的 CPU 周期,验证的程序有多个不同的实例(同一程序、不同输入),这些和数据并行的云计算场景十分吻合,因而研究可验证计算协议,对于解决引言中提出的云计算中的程序执行的可信问题从而构建可信的云计算是有意义的.

然而,可验证计算协议领域以及其构建可信云计算领域在以下几个方面还有待进一步的研究:

(1)相关的理论工具还有待于进一步改进.

一方面需要通过理论工具的研究和改进来降低验证者和证明者的开销,尤其是要把证明者的开销降低到一个合理的范围,使得协议真正能用于实际的场景中.验证者的开销可以分为固定的开销(可以分摊,通常指每个程序或者每次批处理的初始阶段的开销)和可变的开销(程序的每个实例的验证开

销).研究如何降低验证者的可变开销,研究能否使用密码学操作和复杂的理论工具来降低或取消验证者初始阶段的开销.改进基于无预处理的论证系统的可验证计算协议,使得其能用于实际场景.

另一方面,研究利用理论工具来建立更加合理的计算模型,用于高效的表示通用程序,从而提高协议的效率.目前的系统要不不能很好的处理循环结构,要不就是编译的代价太高无法实用.BCGTV 协议可以处理独立于数据的循环的程序,但是其对于程序转化成特殊的电路模型引入了过大的开销.BCGTV 和 Pantry 协议可以处理包含 RAM 的程序,但是 Pantry 协议对内存操作转换成约束集也引入了过大的开销(目前,在 T 机器运行的一些计算机程序原本需要 T 步,转换成由电路计算远远超过 T 步).有必要改进这两种计算模型使得其既可以很好的处理循环结构和 RAM,又不至于引入过大的开销.或者设计新的更加高效的计算模型来表示通用计算.

(2)在系统和编程语言方面值得研究.

针对已有的电路和约束计算模型,设计定制的高级程序语言,降低程序到计算模型的转化开销.

目前,一些可验证计算协议编译处理和证明系统的工作已经有所交叉,而且很多协议在并行计算中性能更优,由于计算模型的高效转化,可以使得验证的效率提高.所以设计一整套相应的高级语言程序、计算模型、验证机器十分必要.

目前还没有协议在真实的云计算场景中测试,开发适用于云计算实际场景的支持并发、访问控制、合理结构的数据库应用的可验证计算,使得协议支持多用户数据库,才能更好的构建可信的云计算.

(3)改变协议的目标和原则,减少可验证计算协议的限制条件,比如可验证计算协议的无条件假设,即除了密码学假设之外不做任何其他假设.如果假设多个证明者之间不能相互交互、合谋,那么多证明协议相对单证明者的开销则降低很多.实际上,如果假设两个证明者至少有一个是计算正确的,就可以使得很多协议能用于特定构造的场景中.

(4)增加安全相关的属性用于构建可信的云计算.Pinocchio、Pantry 协议说明了在证明者对验证者隐藏询问信息的场景下的简单应用,还有很多地方值得研究,比如说提供隐私相关的其他安全属性,可以用来保护云计算用户的隐私.再比如说公开可验证性的特性使得任何拥有密码的用户都可以验证其可信性,这为第 3 方审计来保证云计算的可信性提供了良好的思路.

可验证计算协议把复杂的密码学和理论计算机科学的研究成果用于实际本身就具有里程碑的意义.基于证据的可验证计算协议是一个趋势,这不仅使理论应用于实际,而且开创了理论计算机新的研究领域.虽然可验证计算协议的性能和在云计算实际场景中的部署还有一定的距离.但是以当前的研究节奏,相信不久的将来,就会有基于证据的可验证计算协议应用到云计算的真实场景中.而且,可验证计算的潜力很大,远远不只是云计算.如果这个领域的研究性能降低到合理的范围,除了验证云计算之外,还有更大的价值.将会有新的方法来构建协议,在任何一个模块为另一个模块执行程序的场景中都可以应用.包括微观层面(micro level),如果 CPU 可以验证 GPU,则可以消除硬件错误;宏观层面(macro level)分布式计算将基于不同的可信假设构建.而且,随着计算能力的增加和计算成本的下降,原本无法实践的协议也可以用于实际场景中.

参 考 文 献

[1] Armbrust M, Fox A, Griffith R, et al. A view of cloud computing. Communications of the ACM, 2010, 53(4): 50-58

[2] Chen Y, Paxson V, Katz R. What's new about cloud computing security? University of California at Berkley, Berkeley USA: Technical Report UCB/EECS-2010-5, 2010

[3] Ko R K L, Jagadpramana P, Mowbray M, et al. TrustCloud: A framework for accountability and trust in cloud computing//Proceedings of the 2nd IEEE World Congress on Services. Washington, USA, 2011: 584-588

[4] Sailer R, Zhang X, Jaeger T, van Doorn L. Design and implementation of a TCG-based integrity measurement architecture//Proceedings of the 13th USENIX Security Symposium. San Diego, USA, 2004: 223-238

[5] Sadeghi A-R, Schneider T, Winandy M. Token-based cloud computing: Secure outsourcing of data and arbitrary computations with lower latency//Proceedings of the 3rd Conference on Trust and Trustworthy Computing. Berlin, Germany, 2010: 417-429

[6] Parno B, McCune J M, Perrig A. Bootstrapping Trust in Modern Computers. Germany: Springer, 2011

[7] Seshadri A, Luk M, Shi E, et al. Pioneer: Verifying integrity and guaranteeing execution of code on legacy platforms//Proceedings of the ACM Symposium on Operating Systems Principles(SOSP). Brighton, UK, 2005: 1-15

[8] Anderson D P, Cobb J, Korpela E, et al. SETI@home: An experiment in public-resource computing. Communications of the ACM, 2002, 45(11): 56-61

[9] Castro M, Liskov B. Practical Byzantine fault tolerance and proactive recovery. ACM Transactions on Computer Systems, 2002, 20(4): 398-461

[10] Haeberlen A, Kouznetsov P, Druschel P. PeerReview: Practical accountability for distributed systems//Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP). Stevenson, USA, 2007: 175-188

[11] Malkhi D, Reiter M. Byzantine quorum systems. Distributed Computing, 1998, 11(4): 203-213

[12] Canetti R, Riva B, Rothblum G. Practical delegation of computation using multiple servers//Proceedings of the ACM Conference on Computer and Communications Security (CCS). Chicago, USA, 2011: 17-21

[13] Ateniese A, Burns R, Reza J, Joseph C. PDP: Provable data possession at untrusted stores//Proceedings of the 14th ACM Conference on Computer and Communications Security. Alexandria, USA, 2007: 598-609

[14] Juels A, Kaliski B. Pors: Proofs of retrievability for large files//Proceedings of the 14th ACM Conference on Computer and Communications Security. Alexandria, USA, 2007: 584-597

[15] Babai L. Trading group theory for randomness//Proceedings of the 17th Annual ACM Symposium on the Theory of Computing (STOC). New York, USA, 1985: 421-429

[16] Lund C, Fortnow L, Karloff H, Nisan N. Algebraic methods for interactive proof systems. Journal of the ACM, 1992, 39(4): 859-868

[17] Shamir A. IP=PSPACE. Journal of the ACM, 1992, 39(4): 869-877

[18] Goldwasser S, Kalai Y T, Rothblum G N. Delegating computation: Interactive proofs for muggles//Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC). Victoria, Canada, 2008: 113-122

[19] Goldwasser S, Micali S, Rackoff C. The knowledge complexity of interactive proof systems//Proceedings of the 17th Annual ACM Symposium on Theory of Computing (STOC). New York, USA, 1985: 291-304

[20] Arora S, Lund C, Motwani R, et al. Proof verification and the hardness of approximation problems. Journal of the ACM, 1998, 45(3): 501-555

[21] Arora S, Safra S. Probabilistic checking of proofs: A new characterization of NP. Journal of the ACM, 1998, 45(1): 70-122

[22] Ishai Y, Mahmoody M, Sahai A. On efficient zero-knowledge PCPs//Proceedings of the 9th IACR Theory of Cryptography Conference (TCC). Taormina, Italy, 2012: 151-168

[23] Arora S. Probabilistic Checking of Proofs and Hardness of Approximation Problems [Ph. D. dissertation]. University of California, Berkeley, USA, 1994

[24] Brassard G, Chaum D, Crépeau C. Minimum disclosure proofs of knowledge. Journal of Computer and System Sciences, 1988, 37(2): 156-189

[25] Gennaro R, Gentry C, Parno B. Non-interactive verifiable computing: Outsourcing computation to untrusted workers//Proceedings of the International Association for Cryptologic Research (IACR) International Cryptology Conference (CRYPTO). Santa Barbara, USA, 2010: 465-482

[26] Gentry C, Wichs D. Separating succinct non-interactive arguments from all falsifiable assumptions//Proceedings of the 43rd ACM Symposium on the Theory of Computing (STOC). San Jose, USA, 2011: 99-108

[27] Kilian J. A note on efficient zero-knowledge proofs and arguments (extended abstract)//Proceedings of the 24th ACM Symposium on the Theory of Computing (STOC). New York, USA, 1992: 723-732

[28] Gennaro R, Gentry C, Parno B, Raykova M. Quadratic span programs and succinct NIZKs without PCPs//Proceedings of the 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques. Athens, Greece, 2013: 626-645

[29] Bitansky N, Canetti R, Chiesa A, Tromer E. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again//Proceedings of the 3rd Innovations in Theoretical Computer Science (ITCS). Cambridge, USA, 2012: 326-349

[30] Ishai Y, Kushilevitz E, Ostrovsky R. Efficient arguments without short PCPs//Proceedings of the 22nd Annual IEEE Conference on Computational Complexity (CCC). San Diego, USA, 2007: 278-291

[31] Gentry C, Halevi S, Smart N. Homomorphic evaluation of the AES circuit//Proceedings of the 32nd Annual Cryptology Conference (CRYPTO). Santa Barbara, USA, 2012: 850-867

[32] Chung K-M, Kalai Y, Vadhan S. Improved delegation of computation using fully homomorphic encryption//Proceedings of the 30th Annual Cryptology Conference. Santa Barbara, USA, 2010: 483-501

[33] Atallah M J, Frikken K B. Securely outsourcing linear algebra computations//Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS). Beijing, China, 2010: 48-59

[34] Benabbas S, Gennaro R, Vahlis Y. Verifiable delegation of computation over large datasets//Proceedings of the IACR 31st International Cryptology Conference (CRYPTO). California, USA. 2011: 111-131

[35] Boneh D, Freeman D M. Homomorphic signatures for polynomial functions//Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT). Tallinn, Estonia, 2011: 149-168

[36] Fiore D, Gennaro R. Publicly verifiable delegation of large polynomials and matrix computations, with applications//Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS). Raleigh, USA, 2012: 501-512

[37] Golle P, Mironov I. Uncheatable distributed computations//Proceedings of the RSA Conference 2001. San Francisco, USA, 2001: 425-440

[38] Papamanthou C, Shi E, Tamassia R. Signatures of correct computation//Proceedings of the IACR 10th Theory of Cryptography Conference (TCC). Tokyo, Japan, 2013: 222-242

[39] Sion R. Query execution assurance for outsourced databases//Proceedings of the 31st Conference in the Series of the Very Large Data Bases Conferences (VLDB). Trondheim, Norway, 2005: 601-612

[40] Thompson B, Haber S, Horne W G, et al. Privacy-preserving computation and verification of aggregate queries on outsourced databases//Proceedings of the 9th Privacy Enhancing Technologies Symposium (PETS 2009). Seattle, USA, 2009: 185-201

[41] Wang C, Ren K, Wang J. Secure and practical outsourcing of linear programming in cloud computing//Proceedings of the 30th IEEE International Conference on Computer Communications (IEEE INFOCOM 2011). Shanghai, China, 2011: 820-828

[42] Malkhi D, Nisan N, Pinkas B, Sella Y. Fairplay—A secure two-party computation system//Proceedings of the 13th USENIX Security Symposium. San Diego, USA, 2004: 287-302

[43] Braun B. Compiling computations to constraints for verified computation. The University of Texas at Austin, Austin, USA: Report HR-12-10, 2012

[44] Cook S A. The complexity of theorem proving procedures//Proceedings of the 3rd Annual ACM Symposium on Theory of Computing. New York, USA, 1971: 151-158

[45] Coelho D R. The VHDL Handbook. 1989 Edition. USA: Springer, 1989

[46] Thomas D E, Moorby P R. The Verilog hardware Description Language. Boston, USA: Kluwer Academic Publishing, 1991

[47] Bergamaschi R, Damiano R, Drumm A, Trevillyan L. Synthesis for the "90s": Highlevel and logic synthesis techniques//Proceedings of the ICCAD93 Tutorial Notes. Santa Clara, USA, 1993: 261-269

[48] Galloway D. The transmogrifier C hardware description language and compiler for FPGAs//Proceedings of IEEE symposium of FPGAs for Custom Computing Machines. Napa Valley, USA, 1995: 136-144

[49] Gokhale M, Gomersall E. High level compilation for fine grained FPGAs//Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines. Napa, USA, 1997: 165-173

[50] Peterson K B, O'Connor R B, Athanas P M. Scheduling and partitioning ANSI-C programs onto multi-FPGA CCM architectures//Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines. Napa Valley, USA, 1996: 178-187

[51] Yamauchi T, Nakaya S, Kajihara N. SOP: A reconfigurable massively parallel system and its control-data-flow based compiling method//Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines. Napa Valley, USA, 1996: 148-156

[52] Malkhi D, Nisan N, Pinkas B, et al. The Fairplay project//Proceedings of the 13th USENIX Security Symposium. San Diego, USA, 2004: 287-302

[53] Cormode G, Mitzenmacher M, Thaler J. Practical verified computation with streaming interactive proofs//Proceedings of the 3rd Innovations in Theoretical Computer Science (ITCS) Conference. Massachusetts, USA, 2012: 289-312

[54] Thaler J, Roberts M, Mitzenmacher M, Pfister H. Verifiable computation with massively parallel interactive proofs//Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Computing (HotCloud'12). Berkeley, USA, 2012: 22-28

[55] Vu V, Setty S, Blumberg A J, Walfish M. A hybrid architecture for interactive verifiable computation//Proceedings of the 2013 IEEE Symposium on Security and Privacy (SP'13). Washington, USA, 2013: 223-237

[56] Setty S, McPherson R, Blumberg A J, Walfish M. Making argument systems for outsourced computation practical (sometimes)//Proceedings of the 19th Annual Network & Distributed System Security Symposium. San Diego, USA, 2012: 202-222

[57] Setty S, Vu V, Panpalia N, et al. Taking proof-based verified computation a few steps closer to practicality//Proceedings of the 21st USENIX Security Symposium. Washington State, USA, 2012: 253-268

[58] Setty S, Braun B, Vu V, et al. Resolving the conflict between generality and plausibility in verified computation//Proceedings of the 8th ACM European Conference on Computer Systems. New York, USA, 2013: 71-84

[59] Parno B, Gentry C, Howell J, Raykova M. Pinocchio: Nearly practical verifiable computation//Proceedings of the 34th IEEE Symposium on Security and Privacy. San Francisco, USA, 2013: 238-252

[60] Braun B, Feldman A, Setty S, et al. Verifying computations with state//Proceedings of the 24th ACM Symposium on Operating Systems Principles. Pennsylvania, USA, 2013: 341-357

[61] Ben-Sasson E, Chiesa A, Genkin D, et al. SNARKs for C: Verifying program executions succinctly and in zero knowledge//Proceedings of the 33rd Annual Cryptology Conference. Santa Barbara, USA, 2013: 90-108

[62] Blum M, Evans W, Gemmell P, et al. Checking the correctness of memories//Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS). San Juan, Puerto Rico, 1991: 90-99

[63] Fu K, Kaashoek M F, Maziéres D. Fast and secure distributed read-only file system//Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI). San

Diego, USA, 2000: 1-24

[64] Li J, Krohn M N, Maziéres D, Shasha D. Secure untrusted data repository (SUNDR)//Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI). San Francisco, USA, 2004: 121-136

[65] Merkle R C. A digital signature based on a conventional encryption function//Proceedings of the International Cryptology Conference (CRYPTO 1987). California, USA, 1987: 369-378

[66] Bitansky N, Chiesa A, Ishai Y, et al. Succinct non-interactive arguments via linear interactive proofs//Proceedings of the 10th Theory of Cryptography Conference (TCC 2013). Tokyo, Japan, 2013: 315-333

[67] Ben-Sasson E, Chiesa A, Genkin D, Tromer E. Fast reductions from RAMs to delegatable succinct constraint satisfaction problems//Proceedings of the 4th Innovations in Theoretical Computer Science (ITCS) Conference. Berkeley, USA, 2013: 401-414

[68] Ben-Sasson E, Goldreich O, Harsha P, et al. Short PCPs verifiable in polylogarithmic time//Proceedings of the 20th Annual IEEE Conference on Computational Complexity (CCC). Washington, USA, 2005: 120-134

[69] Ben-Sasson E, Goldreich O, Harsha P, et al. Robust PCPs of proximity, shorter PCPs and applications to coding//Proceedings of the 34th ACM Symposium on Theory of Computing (STOC 2013). Chicago, USA, 2004: 889-974

[70] Ben-Sasson E, Sudan M. Short PCPs with polylog query complexity. SIAM Journal on Scientific Computing, 2008, 38(2): 551-607

[71] Dinur I. The PCP theorem by gap amplification. Journal of the ACM, 2007, 54(3): 735-750

[72] Goldwasser S, Sipser M. Private coins versus public coins in interactive proof systems//Proceedings of the 18th ACM Symposium on Theory of Computing. Boston, USA, 1986: 59-68

[73] Kilian J. Improved efficient arguments (preliminary version)//Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology. Santa Barbara, USA, 1995: 311-324

[74] Setty S. Toward Practical Argument Systems for Verifiable Computation[Ph. D. dissertation]. The University of Texas at Austin, Austin, 2014

[75] Goldreich O. Foundations of Cryptography: Volume 1, Basic Tools. UK: Cambridge University Press, 2001

[76] Gentry C. A Fully Homomorphic Encryption Scheme[Ph. D. dissertation]. California, USA: Stanford University, 2009

[77] Bellare M, Coppersmith D, Hastad J, et al. Linearity testing in characteristic two. IEEE Transactions on Information Theory, 1996, 42(6): 1781-1795

[78] Blum M, Luby M, Rubinfeld R. Self-testing/correcting with applications to numerical problems. Journal of Computer and System Sciences, 1993, 47(3): 549-595

[79] Cachin C. Integrity and consistency for untrusted services//Proceedings of the 37th International Conference on Current Trends in Theory and Practice of Computer Science. Nový Smokovec, Slovakia, 2011: 1-14

[80] Cachin C, Keidar I, Shraer A. Fail-aware untrusted storage. SIAM Journal on Scientific Computing, 2011, 40(2): 493-533

[81] Mahajan P, Setty S, Lee S, et al. Depot: Cloud storage with minimal trust//Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation. Vancouver, Canada, 2010: 307-322

[82] Goldreich O, Goldwasser S, Halevi S. Collision-free hashing from lattice problems. Electronic Colloquium on Computational Complexity (ECCC), 1996, 3(42): 236-241

[83] Ajtai M. Generating hard instances of lattice problems//Proceedings of the ACM Symposium on the Theory of Computing (STOC). Pennsylvania, USA, 1996: 99-108

WANG Jia-Hui, born in 1983, Ph. D. candidate. Her current research interests include cloud computing and cloud security, data security and data protection.

LIU Chuan-Yi, born in 1982, Ph. D., assistant professor. His research interests include cloud computing and cloud security, data security and data protection.

WANG Guo-Feng, born in 1988, Ph. D. candidate. His current research interest is cloud security.

FANG Bin-Xing, born in 1960, Ph. D., professor, Member of Chinese Academy of Engineering. His current research interests include information and network security, content security.

Background

This paper belongs to Cloud Computing and Cloud Security area. How can the client trust results computed by the cloud service provider, or the integrity of data stored by the cloud service provider? This is a classic question in the context of cloud computing. To a very great extent, trust worthiness is a critical factor for the large-scale use of cloud services.

Various solutions have been proposed to solve this problem

including replicate computation, auditing, trusted hardware and attestation. All of approaches must make assumptions about either the class of computations or the failure modes of the performing computers, etc. However, verifiable computation tells us a fully general solution that makes no assumptions about the cloud provider. The client can check the correctness of the computation executed by remote cloud provider via inspecting a proof returned by cloud service provider without reexecuting the computation.

The goal of this paper is to survey a constructive approach to the trusted cloud based on verifiable computation. Our focus on executing verification is motivated by cloud computing more generally. In this scenario, the causes of incorrect execution include corruption of input data in storage or transit, platform bugs, hardware faults, software misconfiguration, etc. Generally speaking, the related works can be divided into two categories. The first approach constructs verifiable computation protocol based on interactive proof system. This approach has the advantage of not using cryptographic protocol and pre-processing phase. So it provides good performance in its special application areas. The second approach constructs verifiable computation protocol based on argument system. This approach can be further divided into two categories, and one of them which does not include pre-processing is still impractical.

This paper presents the process of verifiable computation protocols, and reviews the state-and-art works on this field. This paper also covers basic definition, principle and process of typical protocols, application scenarios, performance analysis, open questions. Though there is a great deal of work remaining to bring verifiable computation to real application of cloud computing, we believe that real application of verifiable computation to cloud computing will appear in the next few years and all our works will play a role in promoting the further research of cloud computing security based on verifiable computation.

This paper is supported by the National Natural Science Foundation of China under Grant No. 61202081: "Study on the Trustworthiness of Cloud Providers", which researches the construction and audit for the tenant trusted execution environment. The group has been working on the trusted cloud computing, cloud storage security, and data privacy. Many papers have been published in respectable international conferences.