doi :10.3969/j.issn.1671-1122.2013.09.005

多租赁用户模型下有效安全外包计算

刘锵[1]，唐春明[2,3]，胡杏[2,3]，张永强[1]

（1. 广东数字证书认证中心有限公司，广东广州 510000；2. 广州大学数学与信息科学学院，广东广州 510006；
3. 广东数学与交叉科学省普通高校重点实验室（广州大学），广东广州 510006）

摘 要：外包计算是云计算环境中常见的服务之一，通常有单租赁用户下的外包计算和多租赁用户下的外包计算两种形式。文章使用安全多方计算协议和密钥共享协议构造多租赁用户模型下的外包计算协议，并且证明了只要多方计算协议和密钥共享方案是安全的，则外包计算协议也是安全的。文章分别为门限和非门限两种情形构造了安全外包计算协议，前者假设每个租赁用户的可信度相同，而后者假设每个租赁用户的可信度不同。

关键词：云计算；外包计算；安全多方计算；密钥共享方案

中图分类号：TP309　　**文献标识码：**A　　**文章编号：**1671-1122（2013）09-0017-05

Efficient and Secure Outsourced Computation in a Multi-tenant Cloud

LIU Qiang[1], TANG Chuan-ming[2,3],HU Xing[2,3],ZHANG Yong-qiang[1]

(1.*Guangdong Certificate Authority CO., LTD. Guangzhou Guangdong* 510010, *China*;2.*School of Mathematics and Information Science, Guangzhou University, Guangzhou Guangdong* 510006, *China*;3.*Key Laboratory of Mathematics and Interdisciplinary Sciences of Guangdong Higher Education Institutes, Guangzhou University, Guangzhou Guangdong* 510006, *China*)

Abstract: Outsourced computation is one of common services in cloud computing, which usually happens in a single-tenant cloud or in a multi-tenant cloud. This paper uses secure multi-party computation and secret sharing scheme as tools to construct outsourced computation in a multi-tenant cloud. The outsourced computation is secure under the assumption that both multi-party computation and secret sharing scheme are secure. This paper constructs threshold outsourced computation and non-threshold outsourced computation in a multi-tenant cloud, where it is threshold if every tenant has identical credibility, otherwise, it is non-threshold.

Key words: cloud computing; outsourced computation; secure multi-party computation; secret share scheme

0 引言

云计算（cloud computing）是用户通过网络以按需付费的方式获得所需的服务和资源的一种新兴的计算方式。有了这种方式，资源限制型（包括数据处理能力、存储空间等）用户（client）可以将计算 / 存储密集型任务委托给资源充裕的云服务器来完成。传统的计算方式中，用户需要自己建立相应的基础设施来完成全部计算，但云用户采用云计算的方式，就可以很大程度地缩减他们的资本支出，减轻云用户端的工作负载，提高效率。

在云计算环境中，终端用户通常只有一个输入和接收设备，几乎没有存储、计算和处理数据的能力。当用户需要执行一个任务，但是没有合适的计算能力去执行它的时候，就需要使用外包计算服务（见图 1）。

图1 外包计算

也就是说，用户不得不委托第三方或者租赁用户（tenant）[1]来执行这项任务。非正式地说，安全的外包计算就是不信任的租赁用户没有用户的输入和输出的任何信息。假设用户有一个输入 x，他需要委托不信任的租赁用户计算函数 f。对于安全的外包计算就是保证：1）秘密性，即 x 和 $f(x)$ 是秘密的，只有用户知道；2）正确性，即不信任的租赁用户给出的结果 $f(x)$ 是正确的；3）可验证性，用户能够验证 $f(x)$ 是否正确；4）有效性，即租赁用户计

收稿日期：2013-08-10

基金项目：国家自然科学基金 [10871222]、广东省高层次人才基金 [201213025]、广东省自然科学基金 [S2012010009950]、广州市教育局基金 [2012A004]

作者简介：刘锵（1972-），男，四川，工程师，硕士，主要研究方向：信息安全；唐春明（1972-），男，湖南，教授，博士生导师，博士，主要研究方向：云计算、信息安全、密码学；胡杏（1981-），女，湖南，博士研究生，主要研究方向：云计算；张永强（1976-），男，广东，高级工程师，博士，主要研究方向：云计算、信息安全、密码学。

万方数据

算 $f(x)$ 的时间应该不超过用户计算 $f(x)$ 的时间。

用户可以委托一个不信任的租赁用户来进行计算 $f(x)$，也可以委托多个不信任的租赁户来进行计算 $f(x)$，前者我们称为单租赁用户模型，后者则称为多租赁用户模型。本文主要研究在多租赁用户模型下怎么实现外包计算。单租赁用户模型下外包计算的实现非常困难，而且都是基于某些假设。

1）相关工作

在单租赁用户模型下主要是进行两类研究。

（1）对于一些具体问题的外包计算研究。文献 [2-11] 研究了关于模指数的外包计算。这些文章的目标是寻求一个（不信任）服务器帮助下的公钥算法协议，能够为一些低消耗的密码处理器（如智能卡）快速的生成一些签名（如 RSA 签名）。已经提出几个协议来解决这些问题，但是有些已经被攻破了。文献 [2] 提出一个协议来外包 RSA 密钥的生成，文献 [12] 改进了这个协议，使得它能抵抗碰撞攻击。外包计算密钥加密和解密协议由 Blaze 和 Luck 提出 [13, 14]。文献 [15] 使用数学隐藏方法讨论了一些安全的外包科学计算（矩阵乘积、偏微分方程、串的匹配等）。

（2）对于一般的函数 $f(x)$，怎样安全地外包计算通常有如下几种方法：①使用完全同态加密方案 [16,17]。该方案最早由 Gentry 提出 [18]，后来该方案被改进或基于其他工具的完全同态加密方案（如格）等被提出 [18,19]，但是这些方案都不是有效的 [20]。如果有效的完全同态加密方案能被构造出来，则所有的外包计算都能使用它计算，研究其他方法也就没有意义。②使用可验证计算的外包计算 [21-25]。主要是使用零知识证明、CS 证明 [24] 以及姚的双方计算协议 [26]。但是这些方案都需要使用完全同态加密方案，因此也不是有效的。③使用辅助卡（token）的外包计算 [27-29]。这些协议都是假设用户有一个辅助卡（如智能卡），这些辅助卡能完成某些指定的任务。但是，这个辅助卡一般由云计算服务商提供，因此不能让用户完全信任，而且它容易受到物理攻击。

在多租赁用户模型下关于外包计算的工作不多。在这种模型下，Kamara 和 Raykova 首次提出了使用安全多方计算为一般函数构造安全外包多方计算 [30] 的思想，但我们发现他们提出的外包计算是不安全的。另外，在文献 [6] 中，Hohenbergera 和 Lysyanskaya 也在多租赁用户模型下（即有两个租赁用户）为模指数计算设计了安全外包计算协议。

对于一般的函数 $f(x)$，在单租赁用户模型下实现外包计算比较困难，因此，本文主要考虑在多租赁用户模型下怎样实现对函数 $f(x)$ 的外包计算。

2）本文结构

第 1 部分介绍本文所需要的一些基本工具。第 2 部分分析文献 [30] 中外包计算的不安全因素，然后构造出正确的外包计算协议。第 3 部分使用信任度构造存取结构，并在满足该存取结构的前提下为任意函数构造一种多租赁用户模型下的安全外包计算协议。

1 基本工具

本文中使用 $x \leftarrow X$ 表示从一个分布 X 中随机取样元素 x。算法 A 的输出 x 用符号 $x \leftarrow A$ 表示。向量 $\vec{v}$ 的第 i 个分量记为 v_i。k 是安全参数。对于函数 $v: N \to N$，如果任意的正多项式 $p(\cdot)$ 和充分大的 k 都有 $v(k) < 1/p(k)$，则 v 是可忽略的。

我们用 $f:\{0,1\}^* \times (\{0,1\}^*)^r \times \{0,1\}^* \to \{0,1\}^*$ 来表示一个 n 方随机函数，其中第一个输入是安全参数 k，第二个输入是一串向量 $\vec{x}$，第三个输入是随机数 r，输出为向量串 $\vec{y}$，这里 $\vec{y} \leftarrow f(k,\vec{x},r)$，$r \leftarrow \{0,1\}^{poly(k)}$。在多方计算环境中，每一方 P_i 拥有一个输入 x_i，他期望能接收到输出结果 y_i。本文将忽略安全参数和随机数，简单记作 $\vec{y} \leftarrow f(\vec{x})$。当函数 $f(k,\vec{x},r)$ 没有输入 r 时，它就是一个确定性函数。假如每一方收到的输出相同，则这个多方函数被称为对称的。

因为任意计算确定性函数的协议都能够用来计算随机函数 [31]，因此，我们只讨论确定性函数。

1.1 密钥共享方案[32]

一个密钥共享方案由两个多项式时间算法 $\Sigma=(Share,Recover)$ 组成。算法 *Share* 把某个输入 x 转变为 n 个秘密分块 $(x_1,x_2,...,x_n)$。在算法 *Recover* 中，任意有资格的参与者集合能够使用他们的秘密分块重构出秘密 x。如果任意有资格的参与者集合能够使用他们的秘密分块重构出秘密 x，则 Σ 是正确的。如果任意非资格的参与者集合都得不到秘密 x 的任何信息，则 Σ 是隐藏的。

1.2 安全多方计算[31]

本节我们使用理想情形和真实协议给出安全多方计算的定义，即通过比较计算函数 f 的真实协议与存在可信第三方的理想协议的输出来定义。

在安全多方计算中，一个攻击者通常控制多个参与者。攻击者分为两类，一类是被动 (passive) 攻击者，一类是主动 (active) 攻击者。被动攻击者仅仅知道它所控制的参与者的状态，而主动攻击者完全控制参与者，甚至可以不按协议执行。另外，如果攻击者在协议执行前就确定收买哪些参与者，则称它为静态（static）攻击者；如果在协议执行过程中还可以选择收买对象，则称它为动态（adaptive）攻击者。在多方计算协议中，由于攻击者的存在，一些攻击行为是不可能被阻止的，如不诚实的参与者选择不参与计算，或者使用任意输入的计算，或者中途退出计算。本文假设攻击者的能力很强，也就是说攻击者是主动的、动态的。

真实协议：在真实协议执行前，每个参与者 p_i 得到它

万方数据

的输入 x_i。对于攻击者 A 来说，如果它是静态的，则它得到收买的参与者集合 $I \subset \{P_1,...,P_n\}$；如果它是动态的，它得到安全参数。在参与者 $P_1,...,P_n$ 和 A 之间真正执行的协议 Π 用 $REAL^{mpc}_{\Pi,A,I}(k,\vec{x})$ 表示，Π 的输出由诚实参与者的输出和 A 的输出构成。

理想情形：假设存在一个可信第三方 T，每个参与者 P_i 把它的输入 x_i 发送给 T，然后由 T 计算函数 $f(x_1,...,x_n)$。那些属于 I 的参与者 P_i 也许发送 x_i（如果攻击者是被动的），也许发送任意的值（如果攻击者是主动的）给 T。如果某个参与者的输入为 $\perp$，T 退出协议发送 $\perp$ 给每个参与者，否则它计算函数 $(y_1,...,y_n) \leftarrow f(x_1,...,x_n)$，$y_i$ 为参与者 P_i 得到的输出。在参与者 $P_1,...,P_n$ 和 A 之间函数 f 的理想计算用 $IDEAL^{mpc}_{f,A,I}(k,\vec{x})$ 表示，它的输出由诚实参与者的输出和 A 的输出构成。

安全性：严格地说，一个计算 f 的协议 Π 是安全的，如果 Π 能仿真理想情形下对 f 的计算。因此，我们有如下正式定义。

定义 1（安全性）：假设 f 是一个 n 方函数，Π 是一个计算 f 的协议。我们说 Π 安全地计算 f，如果对于所有的概率多项式时间的攻击者 A，存在一个概率多项式时间的模拟器 S，使得对于所有的 $I \subset \{P_1,...,P_n\}$，都有 $\{REAL^{mpc}_{\Pi,A,I}(k,\vec{x})\}_{k\in N,\vec{x}\in\{0,1\}^*} \overset{c}{\approx} \{IDEAL^{mpc}_{f,S,I}(k,\vec{x})\}_{k\in N,\vec{x}\in\{0,1\}^*}$，其中 $\overset{c}{\approx}$ 表示计算上不可区分。

也就是说在协议 Π 中，攻击者 A 控制属于 I 的每一个参与者，A 的输出可以用一个模拟器 S 在理想情形下仿真出来[31]。

1.3 安全外包计算[33]

在多租赁用户模型下，安全外包计算允许一个用户在拥有一个秘密输入 x 的情形下安全外包函数 f 的计算到一些不信任的租赁户上。严格地说，一个安全外包方案应该保证：（1）租赁户不能得到用户输入和输出的任何信息；（2）函数能被正确地计算。

下面我们根据文献 [30] 给出多租赁用户模型下安全外包计算的定义。与安全多方计算类似，他们也使用真实协议和理想情形来定义。与安全多方计算定义不同之处在于：（1）在外包计算中，仅仅只有用户提供输入，然而他得到所有的输出；（2）攻击者 A 不能收买用户。

真实协议：在真实协议 Ω 执行过程中，用户拥有秘密输入 x，然而租赁用户没有任何的输入。如果攻击者是静态的，则它在协议执行前选择好它收买的租赁用户集合 $I \subset \{P_1,...,P_n\}$。如果攻击者是动态的，则它在协议执行过程中选择 I。在用户、租赁用户 $P_1,...,P_n$ 和攻击者 A 之间真实执行的协议 Ω 记为 $REAL^{del}_{\Omega,A,I}(k,x)$，它的输出由用户的输出、$P_1,...,P_n$ 的输出、A 的输出构成。

理想情形：在这种情形中，存在一个可信方 T 计算函数 f。与真实协议一样，协议开始时只有用户拥有秘密输入 x，$P_1,...,P_n$ 没有任何输入。如果攻击者 A 是静态的，则它在协议执行前选择好它收买的租赁用户集合 $I \subset \{P_1,...,P_n\}$。如果攻击者 A 是动态的，则它在协议执行过程中选择 I。在用户、租赁用户和攻击者 A 之间执行函数 f 的理想计算，用 $IDEAL^{del}_{f,A,I}(k,x)$ 表示，它的输出由用户的输出、诚实租赁用户的输出和 A 的输出所构成。

安全性：严格地说，一个计算 f 的外包协议 Ω 是安全的，如果 Ω 能仿真理想情形中 f 的计算。它的正式定义如下。

定义 2（安全外包）：假设 f 是一个函数，Ω 是一个计算 f 的外包协议。我们说 Ω 安全地计算 f，如果对于所有的概率多项式时间的攻击者 A，存在一个概率多项式时间的模拟器 S，使得对于所有的 $I \subset \{P_1,...,P_n\}$，都有 $\{REAL^{del}_{\Omega,A,I}(k,x)\}_{k\in N,x\in\{0,1\}^*} \overset{c}{\approx} \{IDEAL^{del}_{f,S,I}(k,x)\}_{k\in N,x\in\{0,1\}^*}$，其中 $\overset{c}{\approx}$ 表示计算上不可区分。

也就是说，任何攻击者在 Ω 中的输出，都可以由理想情形中的一个 S 仿真出来。

2 多租赁用户下的安全外包计算

本节首先分析在文献 [30] 中外包计算协议的缺陷，然后改进它，得到一个安全外包计算协议。这里我们考虑的都是门限的情形。

2.1 Kamara和Raykova的安全外包计算协议

文献 [30] 中，Kamara 和 Raykova 使用安全多方计算协议 Π 和门限密钥共享方案 $\Sigma=(Share,Recover)$ 提出了一个安全外包计算协议 Ω，原理如下：

1）用户根据算法 $Share$ 把输入 x 分成 n 个秘密分块 $x_1,...,x_n$，然后把每个秘密分块 x_i 发送给对应的租赁用户 P_i。

2）定义函数 $f'((x_1,r_1),...,(x_n,r_n))=Share(f(Recover(x_1,...,x_n)),n;r_1\oplus...\oplus r_n)$。租赁用户 $P_1,...,P_n$ 执行安全多方计算协议 Π 去计算 f'。执行的结果是每个租赁用户 P_i 得到输出 $y=f(x)$ 的一个秘密分块 y_i。

3）每个租赁用户 P_i 把秘密分块 y_i 发送给用户后，用户再使用算法 $Recover$ 从 $y_1,...,y_n$ 恢复出 y。

对于上述协议，文献 [30] 给出了如下的结论：如果 Π 在动态和主动攻击者下是安全的，并且 Σ 是一个安全的密钥共享方案，则协议 Ω 在动态和主动攻击者下是安全的。其中 Ω 中攻击者与 Π 中的攻击者是一样的。

对于协议 Ω 存在 3 个如下缺陷：1）在用户和每个租赁用户 P_i 之间应该存在一条秘密通道，否则攻击者可以截取秘密分块恢复出秘密 x 和输出 y；2）仅仅只能处理门限情形，对于非门限情形，文献 [30] 没有考虑；3）最重要的是定义的函数 f' 是不可行的。应该纠正为：

$$f'((x_1,r_1),...,(x_n,r_n))=Share(f(x_1,...,x_n),n;r_1\oplus...\oplus r_n) \quad \cdots(1)$$

如果是原来的 f'，则首先使用算法 $Recover$ 从 $x_1,...,x_n$ 恢复出 x，然后再计算出 $f(x)$，这样会泄露秘密 x。

下节我们将为修改后的 $f'((x_1,r_1),...,(x_n,r_n))=Share(f(x_1,...,x_n),n;r_1\oplus...\oplus r_n)$ 构造

万方数据

门限情形的安全外包计算协议。而在第3部分我们将考虑非门限情形。

2.2 安全的外包计算协议

对于任意函数$f(x)$，假设x能被t个值线性计算，即存在$a_1,...,a_t$，使得$x=a_1x_1+...+a_tx_t$，则函数$f(x)=f(a_1x_1+...+a_tx_t)$。因此，按照Shamir的密钥共享方案，可以把$x$分成$n$个秘密分块$x_1,...,x_n$，其中任意$t$个秘密分块都可以线性计算$x$（$t\leq n$），此时$a_1,...,a_t$可以使用拉格朗日插值公式计算出来。因此，我们能把函数$f(x)$转化为一个n方函数，

$$f_1'((x_1,r_1),...,(x_n,r_n))=Share(f(x_1,...,x_n),n;r_1\oplus...\oplus r_n)\cdots(2)$$

下面给出计算$f(x)$的外包计算协议。

假设用户和每个租赁用户P_i之间都存在一条秘密通道（秘密通道可用物理上的方法实现，也可以用公钥密码协议实现），则我们也使用安全多方计算协议Π和密钥共享方案$\Sigma=(Share,Recover)$为计算$y=f(x)$提出了一个安全外包计算协议Ω_1。

1）用户根据算法$Share$把输入x分成n个秘密分块$x_1,...,x_n$，然后每个秘密分块x_i通过秘密通道被送给P_i（$i=1,...,n$）。

2）$P_1,...,P_n$执行安全多方计算协议Π计算f_1'。执行的结果是P_i得到输出$y=f(x)$的一个秘密分块y_i。

3）P_i把秘密分块y_i通过秘密通道发送给用户后，用户再使用算法$Recover$从$y_1,...,y_n$恢复出y。

因为在协议中，我们假设用户与每个服务器之间都存在秘密通道，因此攻击者想通过截取秘密分块来重构秘密x和y是不可行的。f_1'与f'的本质区别是在f_1'中不用重构x，这样避免了x由于被重构而泄露。

根据秘密共享方案，攻击者收买的租赁用户个数只要不超过门限值，则攻击者就得不到用户的输入和输出任何信息。也就是保证了输入和输出的秘密性。安全多方计算的正确性保证了输出的正确性。可验证性直接从安全多方计算的合理性得到。因此，我们有如下结论：

定理1：如果Π在动态和主动攻击者下是安全的，Σ是一个安全密钥共享方案，则协议Ω_1在动态和主动攻击者下是安全的。其中Ω_1中攻击者与Π中攻击者是一样的。

3 基于信任度的多租赁用户下的安全外包计算

3.1 基于信任度的存取结构

第1部分中我们所使用的存取结构都是门限的，也就是每个租赁用户的能力都是一样的。我们所使用的存取结构如下定义，假设$P=\{P_1,...,P_n\}$，集合$\Gamma\subset 2^P$，其中2^P为P的所有子集构成的集合。Γ满足如下关系，若$A\subseteq P$，且$|A|\geq t$，则$A\in\Gamma$；否则$A\notin\Gamma$。t就是我们说的门限值。

但是，在云计算环境中，没有绝对值得信任的云计算提供商。因此，在现实的“云”中，每个云计算提供商提供的租赁用户信用度是不一样的。对于每一个云计算提供商，假设它们拥有的租赁户以值$\alpha\in[0,1]$作为信用度（该值通过云计算提供商的技术、资产、规模、安全程度等来计算出）。值越大，表示可信程度越强，当$a=1$时，表示是绝对可信任的。事实上，同一个云计算提供商提供的租赁用户的信用度也许也不一样[1]。为了简化，我们假设一样。

根据信用度，我们定义一个非门限存取结构Γ。假设存在n个租赁用户$P=\{P_1,...,P_n\}$，则任何租赁用户集合$A\subseteq P$都属于Γ，只要A中元素的信用度之和≥ 1。

我们定义的存取结构Γ更适合云计算环境，因为不同的云计算提供商由于受到资本、技术的限制，因此他们的可信度不一样。对于特殊的门限存取结构，表明每个云计算提供商的信用程度一样。

3.2 实现非门限存取结构的安全多方计算协议

对于任意的存取结构，为其构造密钥共享方案的方法有好几种[34-37]，但是我们采取文献[36,37]的方法，因为它们能得到实现Γ的最优线性密钥共享方案。这里的最优指的是计算复杂度和通信复杂度是最低的。

根据文献[35]的方法，我们就可以构造实现存取结构Γ的最优安全多方计算协议Π，因为我们的密钥共享方案是最优的。

3.3 安全的非门限外包计算协议

对于任意函数$f(x)$，假设x能被t个值线性计算，即$x=a_1x_1+...+a_tx_t$，则函数$f(x)=f(a_1x_1+...+a_tx_t)$。因此，按照文献[36]中所得到的密钥共享方案，可以把x分成n个秘密分块$x_1,...,x_n$，对于任意的$A\in\Gamma$，$|A|$个秘密分块都可以线性计算x。因此，我们能把函数$f(x)$转化为一个n方函数，$f_1'((x_1,r_1),...,(x_n,r_n))=Share(f(x_1,...,x_n),n;r_1\oplus...\oplus r_n)$，且任意的$A\in\Gamma$都能计算该函数。下面给出计算$f(x)$的外包计算协议。

假设用户和每个P_i之间都存在一条秘密通道，任意可计算函数在n个租赁用户$P_1,...,P_n$中存在一个安全多方计算协议Π以及基于线性码的线性密钥共享方案$\Sigma=(Share,Recover)$，则我们为计算$y=f(x)$提出一个外包计算协议Ω_2，该协议实现的是非门限存取结构。

1）用户根据算法$Share$把输入x分成n个秘密分块$x_1,...,x_n$，每个秘密分块x_i通过秘密通道发送给服务器P_i，需要强调的是x_i也许是一个向量。因为对于任意存取结构Γ，不一定存在理想的密钥共享方案（若P_i的秘密分块具有相同的长度，则它们是理想的密钥共享方案）。

2）租赁用户$P_1,...,P_n$执行安全多方计算协议Π计算f_1'。执行的结果是每个P_i得到输出$y=f(x)$的秘密分块y_i。同理，y_i也

许是一个向量。

3) P_i 把秘密分块 y_i 通过秘密通道发送给用户后，用户再使用算法 Recover 从 $y_1,\ldots,y_n$ 恢复出 y。

对于协议 Ω_2，我们有如下结论：

定理 2：如果 Π 在动态和主动攻击者下是安全的，Σ 是一个安全的密钥共享方案，则协议 Ω_2 在动态和主动攻击者下是安全的。其中 Ω_2 中攻击者与 Π 中攻击者是一样的。

该定理的正确性直接由安全多方计算和密钥共享方案的性质得到。

4 结束语

本文主要解绍了在多租赁用户模型下怎样实现安全外包计算函数 $y=f(x)$。我们使用安全多方计算协议和密钥共享方案为基本工具，分别为门限存取结构情形和非门限存取结构情形构造了安全外包计算协议[38,39]。（责编 马珂）

参考文献

[1]Securing Multi-Tenancy and Cloud Computing (White paper), Juniper Networks, Inc[EB/OL] http://www.juniper.net/us/en/local/pdf/.../2000381-en.pdf,2012-03.

[2]Boneh D, Modadugu N, Kim M. Generating rsa keys on a handheld using an untrusted server[C]. In INDOCRYPT' 00: Proceedings of the first International Conference on Progress in Cryptology, pages 271-282, London, UK, 2000, Springer-Verlag.

[3]Burns J, Mitchell CJ. Parameter selection for server-aided rsa computation schemes[J]. IEEE Trans. Comput. 43(2): 163-174,1994

[4]Dijk M, Clarke D, Gassend B, Suh GE, Devadas S. Speeding up exponentiation using an untrusted computational resource[J]. Des. Codes. Cryptography, 39(2): 253-273, 2006.

[5]Ernvall AM, Nyberg K. On server-aided computation for RSA protocols with private key splitting[J]. In Proceedings of Nordsec 2003. Department of Telematics, NTNU, 2003.

[6]Hohenbergera S, Lysyanskaya A. How to securely outsource cryptographic computations[J]. Lecture Notes in Computer Science(3378), pages 264-282, Springer Berlin/Heidelberg, 2005.

[7]Lim CH, Lee PJ. Security and Performance of server-aided rsa computation protocols[C]. In Proceedings of 15th annual International Cryptology Conference on Advances in Cryptology, pages 70-83. Springer-Verlag, 1995

[8]Matsumoto T, Imai H, Laih CS, Yen CM. On verifiable inplicit asking protocols for rsa computation[C]. In ASIACRYPT' 92: Proceedings of the Workshop on the Theory and Application of cryptographic Techniques, pages 296-307. Springer-Verlag, 1993.

[9]Matsumoto T, Kato K, Imai H. Speeding up secret computations with insecure auxiliary devices[C]. In CRYPTO' 88: Proceedings on Advances in Cryptology, pages 497-506, Springer-Verlag, 1990.

[10]Hong SM, Shin JB, Lee KH, Yoon H. A new approach to server-aided secret computation[C]. In Information Security and Cryptology, pages 33-45, 1998.

[11] Jakobsson M, Wetzel S. Secure server-aided signature generation[C]. In PKC' 01: Proceeding of the 4th International Workshop on practice and Theory in Public key Cryptography, pages 383-401, Springer-Verlag, 2001.

[12] Chen Y, Safavi NR, Baek J, Chen X. Server-aided rsa key generation against collusion attack[C]. Lecture Notes in Computer Science(4074), pages 27-37, Springer Berlin/Heidelberg, 2005.

[13] Blaze M. High-bandwidth encryption with low-bandwidth smartcards[C]. In Proceedings of the Third International Workshop on Fast Software Encryption, pages 33-40, Springer-Verlag, 1996.

[14] Lucks S. On the security of remotely keyed encryption[C]. In FSE' 97: Proceedings of the 4th International Workshop on Fast Software Encryption, pages 219-229, Springer-Verlag, 1997.

[15] Atallah, Pantazopoulos, Rice, Spafford. Secure outsourcing of scientific Computations[C]. In Advances in Computers, Academic Press, volume 54. 2001.

[16] Boneh D, Goh EJ, Nissim K. Evaluating 2-DNF formulas on ciphertexts[C]. In Theory of Cryptography - TCC' 05, pages 325-341, 2005.

[17] Chung KM, Kalai Y, Vadhan S. Improved delegation of computation using fully homomorphic encryption[J]. In Advances in Cryptology-CRYPTO, pages 483-501, 2010.

[18] Gentry C. Fully homomorphic encryption using ideal lattices[J]. In STOC, pages 169-178, ACM, 2009.

[19] Gentry C. Toward basing fully homomorphic encryption on worst-case hardness[J]. In CRYPTO, pages 116-137, 2010.

[20] Lauter K, Naehrig M, Vaikuntanathan V. Can Homomorphic Encryption be Practical[EB/OL]. http://eprint.iacr.org /2011/405, 2011.

[21] Goldwasser S, Kalai YT, Rothblum GN. Delegating computationg: interactive proofs for muggles[C]. In STOC, pages 113-122,2008.

[22] Gennaro R, Gentry C, Parno B. Non-interactive verifiable computing: Outsourcing computation to untrusted workers[C]. In CRYPTO, pages 465-482, 2010.

[23] Chung KM, Kalai Y, Vadhan SP. Improved delegation of computation using fully homomorphic encryption[C]. In CRYPTO, pages 483-501, 2010.

[24] Micali C, Computationally sound proofs, SIAM J. Comput., 30(4): 1253-1298,2000.

[25] Goldwasser S, Lin HJ, Rubinstein A. Delegation of computation without rejection problem from designated verifier CS-proofs[C]. 2011.

[26] Yao A, Protocols for secure computation [C].IEEE FOCS, 160-164, 1982.

[27] Sadeghi AR, Schneider T, Winandy M. Token-Based Cloud Computing[C]. Secure Outsourcing of Data and Arbitrary Computationa with Lower Latency[C]. In TRUST 2010, pages 417-429, 2010.

[28] Jarvinen K, Kolesnikov V, Sadeghi AR, Schneider T. Embedded SFE: Offloading server and network using hardware tokens[C]. In: Financial Cryptography and Data Security (FC2010), 2010.

[29] Trusted Computing Group (TCG). TPM main specification. Main specification[EB/OL].Trusted Computing Group http://www.trustedcomputinggroup.org ,2009.

[30] Kamara S, Raykova M. Secure Outsourced Computation in a Multi-tenant Cloud[EB/OL]. http://www.zurich.ibm.com/~cca/csc2011/submissions/kamara.pdf, 2011.

[31] Goldreich O. The Foundations of Cryptography-Volume 2[M]. Cambridge University Press, 2004.

[32] Shamir A. How to share a secret[C], Communications of the ACM 22(1979), pp. 612-613.

[33] Liu ML, Zhang ZF. Secret Sharing Scheme and Secure Multi-party Computation[M]. Publishing House of Electronics Industry, 2008.

[34] Ito M, Saito A, Nishizeki T. Secret sharing scheme realizing any access structure[C]. Electronics and Communications in Japan (Part III: Fundamental Electronic Science), Vol 72, 56-64, 1987.

[35] Cramer R, Damgard I, Maurer U. General Secure Multi-party Computation from Linear Secret-Sharing Scheme[C]. In EUROCRYPT, pp316-334, 2000

[36] Tang CM, Gao SH, Zhang CL. The Optimal Linear Secret Sharing Scheme for Any Given Access Structure[EB/OL]. http://eprint.iacr.org/2011/147.ps,2011.

[37] Chen YN, Tang CM, Dai SG. The Greatest Lower Bounds of Monotone Span Programs[C].In Proceedings of ChinaCrypt' 2011, pages 159-163, 2011.

[38] 刘木兰，张志芳．密钥共享体制和安全多方计算 [M], 北京：电子工业出版社，2008.

[39] Chen YN, Tang CM, Dai SG. The Greatest Lower Bounds of Monotone Span Programs[C], 中国密码学会论文集 ,2011:159-163.

万方数据