软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(2):399−415 [doi: 10.13328/j.cnki.jos.005585]

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

基于可重随机化混淆电路的可验证计算*

赵青松[1,2,5], 曾庆凯[1,2], 刘西蒙[3,4], 徐焕良[5]

[1](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210023)

[2](南京大学 计算机科学与技术系,江苏 南京 210023)

[3](福州大学 数学与计算机科学学院,福建 福州 350117)

[4](School of Information Systems, Singapore Management University, Singapore 178902, Singapore)

[5](南京农业大学 信息科技学院,江苏 南京 210095)

通讯作者: 曾庆凯, E-mail: zqk@nju.edu.cn

摘 要: Yao 的混淆电路可用于客户端将函数计算外包给服务器,并可验证其正确性.然而,混淆电路仅能使用 1 次.Gennaro 等人组合使用全同态加密和混淆电路,可实现客户端和服务器在多次输入上重用混淆电路.但是,所有已知的全同态加密在效率的提高上似乎仍有很大的空间,并且需要较强的困难性假设.另一方面,Gennaro 等人的方案只能在敌手不能对客户端发起任何数量的验证查询这种较弱的模型下被证明是安全的.部分同态加密的困难性假设要弱于全同态加密,虽然只支持数量有限的同态操作,但比全同态加密运行速度更快、更加紧凑.提出了一个使用加同态加密的可验证计算方案.它基于 DDH 假设,能够容忍任意数量的恶意验证查询,采用的主要技术是可重随机化的混淆电路.该技术可以实现重随机化的混淆电路分布与原有的混淆电路分布在计算上是不可区分的.另外,也给出了一种使用可重随机化的混淆电路构造密码转置防火墙方案,称为可重用密码转置防火墙.也就是说,混淆电路可生成 1 次,接下来,密码转置防火墙可安全地重随机化和重用多次.

关键词: 可验证计算;可重随机化混淆电路;同态加密;密码转置防火墙

中图法分类号: TP309

中文引用格式: 赵青松,曾庆凯,刘西蒙,徐焕良.基于可重随机化混淆电路的可验证计算.软件学报,2019,30(2):399−415. http://www.jos.org.cn/1000-9825/5585.htm

英文引用格式: Zhao QS, Zeng QK, Liu XM, Xu HL. Verifiable computation using re-randomizable garbled circuits. Ruan Jian Xue Bao/Journal of Software, 2019,30(2):399−415 (in Chinese). http://www.jos.org.cn/1000-9825/5585.htm

Verifiable Computation Using Re-randomizable Garbled Circuits

ZHAO Qing-Song[1,2,5], ZENG Qing-Kai[1,2], LIU Xi-Meng[3,4], XU Huan-Liang[5]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210023, China)

[2](Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China)

[3](College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350117, China)

[4](School of Information Systems, Singapore Management University, Singapore 178902, Singapore)

[5](College of Information Science and Technology, Nanjing Agricultural University, Nanjing 210095, China)

Abstract: Yao's garbled circuit allows a client to outsource a function computation to a server with verifiablity. Unfortunately, the garbled circuit suffers from a one-time usage. The combination of fully homomorphic encryption (FHE) and garbled circuits enables the client and the server to reuse the garbled circuit with multiple inputs (Gennaro *et al.*). However, there still seems to be a long way to go

* 基金项目: 国家自然科学基金(61772266, 61572248, 61431008, 61702105)
Foundation item: National Natural Science Foundation of China (61772266, 61572248, 61431008, 61702105)
收稿时间: 2017-11-22; 修改时间: 2018-01-04, 2018-02-05; 采用时间: 2018-04-02; jos 在线出版时间: 2018-04-27
CNKI 网络优先出版: 2018-04-27 14:58:39, http://kns.cnki.net/kcms/detail/11.2560.TP.20180427.1458.011.html

for improving the efficiency of all known FHE schemes and it need much stronger security assumption. On the other hand, the construction is only proven to be secure in a weaker model where an adversary can not issue any number of verification queries to the client. Somewhat homomorphic encryption schemes, whose assumptions are much weaker than the FHE schemes, support a limited number of homomorphic operations. However, they can be much faster and more compact than the FHE schemes. In this work, a verifiable computation scheme is presented which can tolerate any number of malicious verification queries with additively homomorphic encryption. The proposed technique comes from the construction of re-randomizable garbled circuits in which the distribution of the original garbled circuit is computationally indistinguishable from the re-randomized garbled circuit. The proposed scheme is based on the decisional Diffie-Hellman (DDH) assumption. A technique solution is also given to construct cryptographic reverse firewalls, which is called reusable cryptographic reverse firewalls, using re-randomizable garbled circuits. Namely, the solution allows garbled circuits to be generated once and then securely re-randomized for many times on cryptographic reverse firewalls.

Key words: verifiable computation; re-randomizable garbled circuit; homomorphic encryption; cryptographic reverse firewall

1 引 言

可验证计算可使计算能力较弱的客户端将函数计算外包给计算能力强但不被信任的服务器,服务器返回计算结果以及计算正确执行的证据,并要求客户端验证此证据的开支必须比自己重新计算函数的开支要小.另外,可验证计算也需要保证隐私性,即服务器对客户端输入(也可以包括输出)一无所知.Kilian 在早期关于有效论证(argument)[1]和 Micali 的计算正确证明(computationally sound proof,简称 CS Proof)[2]中都提出了计算双方交互只有 1 轮的非交互可验证计算.Gennaro 等人形式化定义和构造了客户端和服务器之间的非交互可验证计算方案[3].随后,在非交互可验证计算方面的许多工作也提出了很多安全外包任意函数的密码方案[4−12].

Yao 的混淆电路(Yao's garbled circuit)是一种实现半诚实敌手下的安全两方计算经典和有效的手段[13,14],但是混淆电路存在电路只能使用 1 次的基本限制,为电路提供超过 1 次的编码输入会损害混淆电路的保密性.Goldwasser 等人采用全同态加密(fully homomorphic encryption,简称 FHE)的方法首次构造了用于两方计算的可重用混淆电路.困难性假设是误差学习(learning with error,简称 LWE)假设,不过,该构造只达到选择(selective)安全[15].Agrawal 也采用 FHE 基于 LWE 假设构造安全性更强的半适应性(semi-adaptive)安全可重用混淆电路[16].在可验证计算背景下,客户端可以采用 Yao 的混淆电路将只有客户端输入的函数外包给不信任的服务器.但是,用于可验证计算的混淆电路也存在电路只能使用一次的基本限制.组合使用 FHE 和混淆电路可使客户端和服务器实现多项式次数输入的混淆电路重用[3].然而,该方案只在敌手不能对客户端发起任何数量的验证查询(verification query)这种较弱的模型下被证明是安全的.同样地,为了满足安全的需要,许多其他的可验证计算解决方案也是依赖于 FHE 的[5,7,9,10].

尽管理论上基于 FHE 的可验证计算是可能的,但实际上因为所用的 FHE 方案效率十分低下[17],所以基于 FHE 的可验证计算构造实际运行开支都很高,且通常需客户端和服务器都执行 FHE,这对于计算能力较弱的客户端,甚至对于计算能力强大的服务器来说,都是很重的负担.

采用密码方法解决可验证计算都需要特定的密码困难性假设.Brakerski 等人提出了基于 LWE 假设的限层全同态加密(leveled fully homomorphic encryption)[18],这是基于格的公钥加密最好的已知困难性假设.但是,所有标准(非限层)FHE 的假设都需要更强的环安全(circular security)假设.因此,我们有兴趣构造可验证计算协议,使其所基于的困难性假设尽可能地弱,这样,如果其中的原语(primitive)被证明对新的攻击是脆弱的,或者新出现的原语具有更好的性能,那么协议所使用的原语就要被替换.

1.1 本文贡献

在本文中,我们将首先构造一个采用加同态加密的非交互可验证计算协议.它是基于 DDH 假设,在任意新输入下能够实现客户端基于混淆电路的外包计算,可以容忍多项式次数的恶意验证查询,并为提供客户端的输入输出隐私保护,以及函数计算的正确性和语义的安全性.

客户端可使用 Yao 的混淆电路,将只有客户端输入的函数计算外包给服务器,但在新输入下重用电路是不

安全的.这是因为电路的输出标签一旦泄露,服务器就可能使用这些标签作为下次外包计算的结果输出.本文解决这个问题的主要方法是可重随机化的混淆电路(re-randomizable garbled circuit),其可实现客户端使用随机数 r 产生的混淆电路,以及服务器根据此混淆电路和客户端给定的随机数 r'(affine transformations,也就是映射转换)产生的重随机化混淆电路(re-randomized garbled circuit),计算能力有限的敌手将不能区分它们[19,20],这意味着原混淆电路和重随机化电路的分布是计算上不可区分的.

然而,由于以下两个因素,可重随机化的混淆电路并不能直接用于可验证计算.

- 首先,服务器产生重随机化混淆电路之后,客户端将函数输入重随机化后发给服务器,服务器就可以根据此重随机化输入逐个门电路(gate by gate)检查重随机化混淆电路,最终得到重随机化的输出.但是,由客户端给定的映射转换和重随机化输入,服务器会获取原混淆电路的有价值信息.
- 其次,重随机化混淆电路的构造过程仅在半诚实模型下是安全的,易受到来自行为可以表现为任何方式的恶意敌手攻击.例如,如果一个恶意的服务器使用同样的映射转换重复重随机化混淆电路,服务器就有可能使用重随机化混淆电路超过 1 次.

重随机化混淆电路过程的有效性证明是典型的零知识(或证据不可区分)证明.一般地,可以使用随机预言机模式经 Fiat-Shamir 范式获取有效的非交互零知识证明[11,20].为了优化针对恶意敌手的协议,本文将采用第 3 节的方法,不使用 Fiat-Shamir 范式,实现恶意敌手下的安全重随机化混淆电路,从而使协议更简洁.

为了能够将可重随机化的混淆电路应用于可验证计算,采用的主要技术手段是数学隐藏方法(mathematical disguise method)[21]和 Kilian 的随机化技术(Kilian's randomization technique)[22].数学隐藏方法使服务器在重复随机化混淆电路过程中不能重用映射转换,以及客户端的开销与函数的计算开支无关.Kilian 的随机化技术随机隐藏映射转换的每个部分,从而确保服务器按序重随机化电路.该技术手段可以抵御混合输入攻击(mixed-input attack)和部分计算攻击(partial evaluation attack).此外,由于上述可验证计算协议是静态(static)安全的,因此,我们还给出了该协议实现适应性(adaptive)安全的方法.

在本文的最后部分,将上述构造应用于密码转置防火墙(cryptographic reverse firewall)[23],给出一种不采用 FHE 安全两方计算下的混淆电路重用方法.密码转置防火墙可被解释为一种状态算法,该算法能够处理安装防火墙的用户发送和接收的已由某些密码算法处理的消息.一方面,如果原有实现已被破坏,则密码转置防火墙将恢复其安全性;另一方面,如果原有实现是正确的,但密码转置防火墙是不安全的,此时密码转置防火墙并不比任何主动敌手更具破坏性.例如,利用某些特定的协议性质,它能够挂起拒绝服务攻击,如丢掉一些消息.从这个角度来说,不必比传播媒介更信任密码转置防火墙.

密码转置防火墙的关键技术是可重随机化的不经意传输(re-randomizable oblivious transfer)和可重随机化的混淆电路.然而,密码转置防火墙重随机化混淆电路超过 1 次是不安全的.为了打破该局限,也是作为上述构造的一个应用,本文提出一种新的密码转置防火墙模型——可重用密码转置防火墙,用户生成混淆电路 1 次,然后可重用转置密码防火墙可安全的重随机化混淆电路多次.

2 预备知识

在下文中,λ 表示安全参数.若一个函数的衰减速度比任意多项式的逆要快,则该函数是可忽略的,用 $negl()$ 来表示.若 n 是一个整数,则$[n]$表示集合$\{1,...,n\}$.$X=\{X_n\}_{n\in\mathbf{N}}$ 和 $Y=\{Y_n\}_{n\in\mathbf{N}}$ 表示两个分布全体,若对所有非一致概率多项式时间 D 和每个 $n\in\mathbf{N}$,$\Pr[D(X_n)=1]-\Pr[D(Y_n)=1]$是可忽略的,则 X 和 Y 是计算上不可区分的,用 $X\overset{C}{\equiv}Y$ 来表示.用小写黑体字母表示向量,大写黑体字母表示矩阵.

2.1 可验证计算定义

遵循文献[3,10],下面介绍非交互可验证计算定义.假设客户端将函数 f 计算外包给服务器,然后服务器返回计算结果和计算正确执行的证据.可验证计算方案$\mathcal{VC}$的形式化描述由以下 4 种算法组成.

- $\boldsymbol{KeyGen}(1^{\lambda}, f)\rightarrow(pk,sk)$. 随机化密码生成算法将安全参数$\lambda$和函数 f 作为输入,生成公钥 pk 和私钥 sk,

客户端将公钥发送给服务器,私钥由客户端秘密保存.

- ***ProbGen***$(sk,x)\rightarrow(\sigma_x,\tau_x)$.问题生成算法将密钥 sk 和客户端的函数输入 x 作为输入,输出 x 的编码输入σ_x和秘密值τ_x.σ_x由客户端发送给服务器用于计算,τ_x由客户端用于验证.
- ***Compute***$(pk,\sigma_x)\rightarrow(\sigma_y)$:给定公钥 pk 和编码输入σ_x,服务器运行该算法计算函数 f 输出 $y=f(x)$的编码形式σ_y.
- ***Verify***$(sk,\sigma_y,\tau_x)\rightarrow(acc,y)$.输入密钥 sk 和秘密值τ_x,客户端执行验证算法.该算法将服务器的编码输出σ_y转换成一个比特 acc 和一个字符串 y.如果 acc=1,客户端就接受计算结果 $y=f(x)$;如果 acc=0,客户端就拒绝接受计算结果.

若恶意的服务器输入由算法 **ProbGen** 生成的σ_x,执行算法 **Compute** 产生的结果不能被验证成功且与函数 f 在输入 x 的计算结果不一致,则该可验证计算方案$\mathcal{VC}$是正确的.

定义 1(正确性). $\forall x\in Domain(f),\forall f\in\mathcal{F}$,其中,$\mathcal{F}$是一个函数族,若 $(pk,sk)\leftarrow$ ***KeyGen***$(1^{\lambda},f)$,$(\sigma_x,\tau_x)\leftarrow$ ***ProbGen***(sk,x),$(\sigma_y)\leftarrow$***Compute***(pk,σ_x),则$(1,y=f(x))\leftarrow$***Verify***(sk,σ_y,τ_x)以不可忽略的概率成立,那么该可验证计算方案$\mathcal{VC}$是正确的.

若敌手不能使验证算法接受一个不正确的输出,则可验证计算方案$\mathcal{VC}$是安全的.也就是说,对于函数 f 和输入 x,若 $\hat{y}\neq f(x)$, 则 **Verify** 不能输出 $\hat{acc}=1$.

考虑关于敌对的服务器$\mathcal{A}$的如下实验.

Experiment $\boldsymbol{Exp}_{\mathcal{A}}^{Verify}[\mathcal{VC},f,\lambda]$

$(pk,sk)\leftarrow$ ***KeyGen***$(1^{\lambda},f)$

For $i=1$ to $\ell=poly(\lambda)$:

$x_i\leftarrow\mathcal{A}^{\boldsymbol{PVerify}(\tau_j,\cdot)}(pk,x_1,\sigma_{x,1},...,x_{i-1},\sigma_{x,i-1})$

$(\sigma_{x,i},\tau_{x,i})\leftarrow$ ***ProbGen***(sk,x_i)

$(i,\hat{\sigma}_y)\leftarrow\mathcal{A}^{\boldsymbol{PVerify}(\tau_j,\cdot)}(pk,x_1,\sigma_{x,1},...,x_{\ell},\sigma_{x,\ell})$

$(\hat{acc},\hat{y})\leftarrow$ ***Verify***$(sk,\hat{\sigma}_y,\hat{\tau}_{x,i})$

If $\hat{acc}=1$ and $\hat{y}\neq f(x_i)$, output 1, else output 0

其中,预言机 ***PVerify***(σ,τ)返回 acc,当且仅当 ***Verify***$(sk,\sigma,\tau)\rightarrow(acc,y)$,$\mathcal{A}^{\boldsymbol{PVerify}}$ 表示敌手$\mathcal{A}$的 **PVerify** 查询.

上述实验中,敌手通过访问预言机生成多个问题实例,并检查客户端对任意编码的响应.若给定一个输入值,敌手产生输出使验证算法接受该错误的输出值,则敌手成功.

定义 2(安全性). 为可验证计算方案$\mathcal{VC}$定义敌手$\mathcal{A}$,其在上述实验中的优势为

$$ADV_{\mathcal{A}}^{Verif}(\mathcal{VC},f,\lambda)=\Pr[\boldsymbol{Exp}_{\mathcal{A}}^{Verif}[\mathcal{VC},f,\lambda]=1].$$

若对所有的概率多项式时间敌手$\mathcal{A}$, $ADV_{\mathcal{A}}^{Verif}(\mathcal{VC},f,\lambda)$ 可以忽略不计是成立的,则验证计算$\mathcal{VC}$是安全的.

输入(输出)隐私的定义采用不可区分方法以确保没有输入(输出)信息泄露.由输入隐私立即得到输出隐私.若有两个不同的输入,问题生成算法 **ProbGen** 产生的两个输出是计算上不可区分的,则可验证计算方案$\mathcal{VC}$是隐私的.实验定义如下.

Experiment $\boldsymbol{Exp}_{\mathcal{A}}^{Priv}[\mathcal{VC},f,\lambda]$

$(pk,sk)\leftarrow$ ***KeyGen***$(1^{\lambda},f)$

$(x_0,x_1)\leftarrow\mathcal{A}^{\boldsymbol{PVerify},\boldsymbol{PProbGen}}(pk)$

$(\sigma_0,\tau_0)\leftarrow$ ***ProbGen***(sk,x_0)

$(\sigma_1,\tau_1)\leftarrow$ ***ProbGen***(sk,x_1)

$b\leftarrow\{0,1\}$

$\hat{b}\leftarrow\mathcal{A}^{\boldsymbol{PVerify},\boldsymbol{PProbGen}}(pk,x_0,x_1,\sigma_b)$

If $\hat{b}=b$, output 1, else output 0

其中,预言机 ***PProbGen***(*x*)表示调用 ***ProbGen***(*sk*,*x*)生成(σ_x,τ_x),且仅返回σ_x,$\mathcal{A}^{\boldsymbol{PProbGen}}$表示敌手$\mathcal{A}$的 **PProbGen** 查询.

定义 3(隐私性). 为可验证计算方案$\mathcal{VC}$定义敌手$\mathcal{A}$,其在上述实验中的优势为

$$ADV_{\mathcal{A}}^{Priv}(\mathcal{VC},f,\lambda)=\Pr[\boldsymbol{Exp}_{\mathcal{A}}^{Priv}[\mathcal{VC},f,\lambda]=1].$$

若对任意的 $f\in\mathcal{F}$和任意的概率多项式时间敌手$\mathcal{A}$, $ADV_{\mathcal{A}}^{Priv}(\mathcal{VC},f,\lambda)-\frac{1}{2}$ 可以忽略不计是成立的,则可验证计算$\mathcal{VC}$是隐私的.

在外包函数 *f* 的每次计算过程中,客户端执行的算法 **ProbGen** 和 **Verify** 共同复杂度要比函数 *f* 的复杂度要小,其中未考虑复杂度为 *poly*(|*f*|)的算法 **KeyGen**.原因是由于考虑的是摊销复杂度模式,即为了提高在线阶段效率,客户端需在离线阶段付出大量的计算开销(和函数 *f* 的复杂度相同).

定义 4(效率). $\forall x\in Domain(f),\forall f\in\mathcal{F}$,其中,$\mathcal{F}$是一个函数族,若算法 **ProbGen** 和算法 **Verify** 共同运行时间是 *o*(*T*),其中,*T* 是计算函数 *f* 所需时间,则可验证计算方案$\mathcal{VC}$是有效率的.

2.2 BHHO加密算法

BHHO 加密算法是一个基于 DDH 假设的公钥加密算法[24].定义 *q* 是群$\mathbb{G}$的阶,*g* 是群$\mathbb{G}$的生成元,$\ell=\lceil 3\log q\rceil$.该公钥加密算法 PKE 由以下 3 种算法组成.

- ***Gen***(1^λ). 从群$\mathbb{G}$和 $\{0,1\}^\ell$ 中分别一致随机选择向量 $(g_1,...,g_\ell)$ 和比特串 $s=(s_1,...,s_\ell)$, 计算 $h=(\prod_{i=1}^{\ell}g_i^{s_i})^{-1}$、密钥 $sk=s$、公钥 $pk=((g_1,...,g_\ell),h)$.
- ***Enc***(*pk*,*m*).随机选择 $r\leftarrow Z_q$.群元素 $m\in\mathbb{G}$加密后的密文形式为 $(g_1^r,...,g_\ell^r,h^r\cdot m)$.
- ***Dec***(*sk*,*c*).密文 $c=(c_1,...,c_{\ell+1})$.算法输出 $m=c_{\ell+1}\cdot\prod_{i=1}^{\ell}c_i^{S_i}$.

BHHO 算法的密钥和明文都具有加同态性质.定义 $f(\boldsymbol{x})=A\boldsymbol{x}+b$ 为从 Z_q^ℓ 到 Z_q^ℓ 的可转置映射转换(invertible affine transformation,简称 IAT).若 $M^{-1}(x^\top|1)^\top=(f(\boldsymbol{x})^\top|1)^\top$, 则定义 M 为 $f(\boldsymbol{x})$的逆映射转换(reverse affine transformation,简称 RAT).若给定 BHHO 公钥 *pk* 和密钥 *sk*,加密比特 *p* 的密文为 $c\in\mathbb{G}^{\ell+1}$,则设密钥 $sk'=f(sk)\in\{0,1\}^\ell$ 是 0-1 向量,那么 $pk\cdot M$ 是 sk'的公钥,$c\cdot M$ 是关于 $pk\cdot M$ 同样比特 *p* 的密文.明文具有同样的同态性质.对于密钥同态,因为在计算过程中用到了转置运算,所以映射转换必须是可转置的.而对于明文同态,任意的映射转换都可以.

2.3 混淆电路

本节介绍混淆电路的构造,采用 Yao 原有构造方法[13,14,19]的表达方式.在半诚实模式下的安全两方计算中,有一对参与方 Alice 和 Bob 有各自的输入,组合使用混淆电路和不经意传输可实现安全计算函数.与此不同的是,在可验证计算中,只有客户端有私有的输入.

设有一系列具有 *n*-bit 输入的布尔电路$\{C_n\}_{n\in\mathbf{N}}$,对于电路 $C\in\{C_n\}_{n\in\mathbf{N}}$中电线 *w*,客户端随机选择两个$\ell$-bit 标签 L_w^0,L_w^1, 分别表示电线 *w* 的输入比特为 0 和 1,其中,ℓ是 BHHO 的密钥长度.给定输入电线分别是 *a* 和 *b*,输出电线为 *c* 的门电路 *g*,为其随机选择 4 个新 2ℓ-bit 的掩码(mask)$\delta_{i,j}$,其中 $i,j\in\{0,1\}$,计算如下 4 个密文对:

$$\{(Enc_{L_a^i}(\delta_{i,j}),Enc_{L_b^j}((L_c^k|0^\ell)\oplus\delta_{i,j})):i,j\in\{0,1\},k=i*j\}\tag{1}$$

其中,操作符*表示门电路的相应操作.客户端使用 BHHO 密钥 L_a^i 加密掩码$\delta_{i,j}$,使用另一个 BHHO 密钥 L_b^j 加密经过与掩码异或的标签(与ℓ-bit 0 连接).4 个密文对随机排序以混淆电路结构.密钥(也就是电线标签)由客户端秘密存放.在电路计算过程中,客户端将整个混淆电路Γ和输入电线的标签(也就是客户端输入 $x\in\{0,1\}^n$ 相应的编码 *c*)发送给服务器,服务器逐门电路检查电路,对于门电路 *g*,服务器获知两根输入电线标签 L_a 和 L_b,用 L_a 解密每个密文对的前半部分,用 L_b 解密其后半部分,并异或它们,如得到 $L_c^k|0^\ell$ 形式,就取其前半部分 L_c 作为门电路输出.最后,服务器计算所有的电路输出电线标签并发送给客户端.

定义 5(混淆电路). $\{C_n\}_{n\in\mathbf{N}}$表示一系列具有 n-bit 输入的布尔电路,电路 $C\in\{C_n\}_{n\in\mathbf{N}}$的混淆电路方案 ***Gb*** 由以下 3 个过程组成.

- ***Gb.Garble***$(1^\lambda,C)\to(\Gamma,gsk)$:获取电路 C,输出混淆电路Γ和密钥 gsk.
- ***Gb.Enc***$(gsk,c)\to c$:获取输入 x 和密钥 gsk,输出编码 c.
- ***Gb.Eval***$(\Gamma,c)\to y$:获取混淆电路Γ和 c,计算输出 y.

定义 6(混淆电路的正确性). 对每个安全参数λ,$\forall C\in\{C_n\}_{n\in\mathbf{N}}$,所有的 $x\in\{0,1\}^n$:

$$\Pr\left[\begin{array}{l}(\Gamma,gsk)\leftarrow Gb.Garble(1^\lambda,C)\\ c\leftarrow Gb.Enc(gsk,x)\\ y\leftarrow Gb.Eval(\Gamma,c):y=C(x)\end{array}\right]=1-negl(\lambda).$$

定义 7(静态安全). 混淆电路是静态安全的,如果存在 PPT 模拟器 S,对任意 PPT 敌手$\mathcal{A}$:

$$\Pr[\boldsymbol{Exp}_{\mathcal{A},s}^{static}(1^\lambda,0)=1]-\Pr[\boldsymbol{Exp}_{\mathcal{A},s}^{static}(1^\lambda,1)=1]\leqslant negl(\lambda).$$

其中, $\boldsymbol{Exp}_{\mathcal{A},s}^{static}(1^\lambda,b)$ 实验定义如下:

1. 挑战者随机选择比特 b.
2. 敌手$\mathcal{A}$向挑战者提交 C 和 x.
3. 如果 b=0,则挑战者生成 $(\tilde{\Gamma},gsk)\leftarrow$ ***Gb.Garble***$(1^\lambda,C)$ 和 $\tilde{c}\leftarrow$ ***Gb.Enc***(gsk,x), 返回 $\tilde{\Gamma}$ 和 $\tilde{c}$.
4. 如果 b=1,则挑战者生成 $(\tilde{\Gamma},state)\leftarrow S(1^\lambda,T(C))$ 和 $\tilde{c}\leftarrow S(C(x),state)$,其中,$T(C)$揭露 C 的拓扑结构,返回 $\tilde{\Gamma}$ 和 $\tilde{c}$.
5. 最后,敌手$\mathcal{A}$输出猜测比特 b',如果 $b'=b$,则敌手$\mathcal{A}$获胜.

定义 8(适应性安全). 混淆电路是适应性安全的,如果存在 PPT 模拟器 S,则对任意 PPT 敌手$\mathcal{A}$:

$$\Pr[\boldsymbol{Exp}_{\mathcal{A},s}^{adaptive}(1^\lambda,0)=1]-\Pr[\boldsymbol{Exp}_{\mathcal{A},s}^{adaptive}(1^\lambda,1)=1]\leqslant negl(\lambda),$$

其中,实验 $\boldsymbol{Exp}_{\mathcal{A},s}^{adaptive}(1^\lambda,b)$ 定义如下:

1. 挑战者随机选择比特 b.
2. 敌手$\mathcal{A}$向挑战者提交 C,挑战者返回 $\tilde{\Gamma}$. 如果 b=0,则挑战者生成 $(\tilde{\Gamma},gsk)\leftarrow$ ***Gb.Garble***$(1^\lambda,C)$; 如果 b=1,则挑战者生成 $(\tilde{\Gamma},state)\leftarrow S(1^\lambda,T(C))$, 其中,$T(C)$揭露 C 的拓扑结构.
3. 敌手$\mathcal{A}$向挑战者提交 x,挑战者返回 $\tilde{c}$. 如果 b=0,则挑战者生成 $\tilde{c}\leftarrow$ ***Gb.Enc***(gsk,x); 如果 b=1,则挑战者生成 $\tilde{c}\leftarrow S(C(x),state)$.
4. 最后,敌手$\mathcal{A}$输出猜测比特 b',如果 $b'=b$,则敌手$\mathcal{A}$获胜.

2.4 可重随机化的混淆电路

Gentry 等人为实现"i-hop"同态加密方案,构造了基于 DDH 假设的可重随机化的混淆电路[19,20],利用 BHHO 的同态性质实现电路的重随机化,将密钥和明文看作向量,它们是关于 Z_q 任意映射函数的同态(我们定义映射函数为与置换矩阵(permutation matrix)相乘).通过两个随机置换π,π',可将密文 $Enc_L(L')$转换成 $Enc_{\pi(L)}(\pi'(L'))$.对于输入电线分别是 a 和 b、输出电线为 c 的门电路 g,等式(1)定义了它的 4 个密文对.RAT π_a,π_b,π_c 分别是与 a,b,c 相对应的$[\ell+1]$上的随机比特序列,采用 BHHO 密钥同态和明文同态性质将等式(1)的 4 个密文对转换为

$$\{(Enc_{\pi'_a(L_a^i)}(\tilde{\pi}'_c(\delta_{i,j})),Enc_{\pi'_b(L_b^j)}(\tilde{\pi}'((L_c^k\,|\,0^\ell)\oplus\delta_{i,j})):i,j\in\{0,1\},k=i*j\}\tag{2}$$

其中, π'_a,π'_b,π'_c 分别是 a,b,c 的 IAT; $\tilde{\pi}'_c(\cdot)$ 表示仅重随机化 2ℓ-bit 参数的前ℓ-bit;转换操作相当于$\pi_a\cdot\pi_c$乘以 4 个密文对前半部分,$\pi_b\cdot\pi_c$ 乘以 4 个密文对后半部分.π_a,π_b 分别是 a,b 用于密钥同态操作的 RAT,π_c用于当前门电路明文同态操作的随机比特序列,但在下一个要检查的门电路中又是用于输入电线之一的密钥同态操作 RAT.为方便起见,将π_a,π_b,π_c在当前门电路中都称为 RAT.每对密文都需随机选择掩码 $\delta'_{i,j}$, 根据 BHHO 的明文同态特性将密文对的前后已加密部分与 $\delta'_{i,j}$ 进行异或处理.

定义 9(可重随机化的混淆电路). π_a,π_b,π_c 是$[\ell+1]$上的随机比特序列 RAT,混淆电路Γ的可重随机化混淆电

路方案由如下 3 个过程组成:***reRandGb***={***reRandGb.Gate***,***reRandGb.InLabel***,***reRandGb.OutLabel***}.

- $reRandGb.Gate(g,\pi_a\cdot\pi_c,\pi_b\cdot\pi_c,\{\delta'_{i,j}\}_{i=0,j=0}^{i=1,j=1})\to\tilde{g}$.输入门电路 g、π_a 与 π_c 乘积、π_b 与 π_c 乘积、随机掩码 $\{\delta'_{i,j}\}_{i=0,j=0}^{i=1,j=1}$, 计算输出重随机化门电路 $\tilde{g}$.
- $reRandGb.InLabel(\pi'_w,L_w^a)\to\pi'_w(L_w^a)$. 设 $a\in\{0,1\}$,输入标签 L_w^a 和 IAT π'_w, 输出重随机化标签 $\pi'_w(L_w^a)$.
- $reRandGb.OutLabel(\pi'_w,\pi'_w(L_w^a))\to L_w^a$. 输入重随机化标签 $\pi'_w(L_w)$ 和 IAT π'_w, 输出标签 L_w.

3 技术概述

文献[19]中构造了基于 BHHO 加密算法(详见第 2.2 节)的可重随机化混淆电路.在本文中,需要服务器利用 BHHO 算法的同态性质和可重随机化性安全重复重随机化来自客户端的混淆电路.具体来说,给定输入电线分别是 a 和 b,输出电线为 c 的门电路 g(用 4 个密文对表示),客户端选择 3 个与电线 a,b,c 分别对应的 RAT π_a,π_b,π_c,以及 4 个随机掩码 $\delta'_{i,j}$, 其中,$i,j\in\{0,1\}$,将它们发送给服务器(我们定义 RAT 为与一个置换矩阵的乘积运算,IAT 也类似).接下来,服务器将每个密文对的前半部分与 π_a,π_b 分别相乘,后半部分与 π_b,π_c 分别相乘,并且将 4 个掩码与此时的 4 个密文对分别做异或运算.但是,服务器直接使用 RAT π_a,π_b,π_c 重随机化门电路会导致混淆电路重用变得不再安全.重随机化标签的方法是从混淆电路的电线标签 L 和其 IAT π' 入手,将 π' 应用于 L,也就是 $\pi'(L)$,而 π' 又可以从随机的 RAT 获得.于是,服务器在逐个门电路检查重随机化电路时,就能从 π' 和 $\pi'(L)$ 中提取标签 L.在环安全中,模拟器已知 RAT 但不知重随机化的密钥[24].

我们的想法是限制服务器明确地获知 RAT 或 IAT,解决方案是使用数学隐藏方法[21]和 Kilian 的随机化技术[22].数学隐藏方法可将每个 RAT 表达为 3 个矩阵的乘法链,Kilian 的随机化技术可将其中的每个矩阵前乘和后乘可转置随机矩阵.例如,设某个 RAT 为可转置随机矩阵 $\boldsymbol{A}$,表示为 3 个矩阵 $\boldsymbol{B},\boldsymbol{C},\boldsymbol{D}$ 乘法链,选取可转置随机矩阵 $\boldsymbol{R}_1,\boldsymbol{R}_2$,将 3 个矩阵分别表示为随机化形式: $\boldsymbol{R}_0\boldsymbol{B}\boldsymbol{R}_1^{-1},\boldsymbol{R}_1\boldsymbol{C}\boldsymbol{R}_2^{-1},\boldsymbol{R}_2\boldsymbol{D}\boldsymbol{R}_0^{-1}$, 其中,$\boldsymbol{R}_0$ 是单位矩阵,它们相乘即可恢复矩阵 $\boldsymbol{A}$.

客户端的随机化密钥生成算法 **KeyGen** 为混淆电路的门电路 i 一致选取可转置随机矩阵 $\boldsymbol{P}_{i,1},\boldsymbol{P}_{i,2},\boldsymbol{P}_{i,3}\in\{0,1\}^{(\ell+1)\times(\ell+1)}$, 为混淆电路随机选取随机矩阵 $\boldsymbol{R}_1,\boldsymbol{R}_2,\boldsymbol{R}_3,\boldsymbol{R}_4\in\{0,1\}^{(\ell+1)\times(\ell+1)}$, 且构建矩阵 $\hat{\boldsymbol{P}}_{i,1}=\boldsymbol{R}_1\boldsymbol{P}_{i,1}\boldsymbol{R}_2^{-1},\hat{\boldsymbol{P}}_{i,2}=\boldsymbol{R}_1\boldsymbol{P}_{i,2}\boldsymbol{R}_2^{-1},\hat{\boldsymbol{P}}_{i,3}=\boldsymbol{R}_3\boldsymbol{P}_{i,3}\boldsymbol{R}_4^{-1}$(由服务器为门电路选取掩码 $\delta'_{i,j}$ 对电路的安全性没有影响).接下来,在每次外包计算中,客户端的问题生成算法 **ProbGen** 也为混淆电路选取可转置随机矩阵 $\boldsymbol{P}_4,\boldsymbol{P}_5\in\{0,1\}^{(\ell+1)\times(\ell+1)}$, 构造 3 个矩阵 $\hat{\boldsymbol{P}}_4=\boldsymbol{R}_0\boldsymbol{P}_4\boldsymbol{R}_1^{-1},\hat{\boldsymbol{P}}_5=\boldsymbol{R}_4\boldsymbol{P}_5\boldsymbol{R}_0^{-1},\hat{\boldsymbol{P}}_{5,4}=\boldsymbol{R}_2\boldsymbol{P}_5\boldsymbol{P}_4\boldsymbol{R}_3^{-1}$, 其中,$\boldsymbol{R}_0$ 是大小合适的单位矩阵.接下来,服务器执行算法 **Compute** 为门电路 i 计算矩阵链乘:

$$\boldsymbol{Z}_{i,1}=\hat{\boldsymbol{P}}_4\hat{\boldsymbol{P}}_{i,1}\hat{\boldsymbol{P}}_{5,4}\hat{\boldsymbol{P}}_{i,3}\hat{\boldsymbol{P}}_5=\boldsymbol{R}_0\boldsymbol{P}_4\boldsymbol{R}_1^{-1}\boldsymbol{R}_1\boldsymbol{P}_{i,1}\boldsymbol{R}_2^{-1}\boldsymbol{R}_2\boldsymbol{P}_5\boldsymbol{P}_4\boldsymbol{R}_3^{-1}\boldsymbol{R}_3\boldsymbol{P}_{i,3}\boldsymbol{R}_4^{-1}\boldsymbol{R}_4\boldsymbol{P}_5\boldsymbol{R}_0^{-1}=\boldsymbol{P}_4\boldsymbol{P}_{i,1}\boldsymbol{P}_5\boldsymbol{P}_4\boldsymbol{P}_{i,3}\boldsymbol{P}_5,$$

$$\boldsymbol{Z}_{i,2}=\hat{\boldsymbol{P}}_4\hat{\boldsymbol{P}}_{i,2}\hat{\boldsymbol{P}}_{5,4}\hat{\boldsymbol{P}}_{i,3}\hat{\boldsymbol{P}}_5=\boldsymbol{R}_0\boldsymbol{P}_4\boldsymbol{R}_1^{-1}\boldsymbol{R}_1\boldsymbol{P}_{i,2}\boldsymbol{R}_2^{-1}\boldsymbol{R}_2\boldsymbol{P}_5\boldsymbol{P}_4\boldsymbol{R}_3^{-1}\boldsymbol{R}_3\boldsymbol{P}_{i,3}\boldsymbol{R}_4^{-1}\boldsymbol{R}_4\boldsymbol{P}_5\boldsymbol{R}_0^{-1}=\boldsymbol{P}_4\boldsymbol{P}_{i,2}\boldsymbol{P}_5\boldsymbol{P}_4\boldsymbol{P}_{i,3}\boldsymbol{P}_5.$$

其中,$\boldsymbol{P}_4\boldsymbol{P}_{i,1}\boldsymbol{P}_5,\boldsymbol{P}_4\boldsymbol{P}_{i,2}\boldsymbol{P}_5$ 和 $\boldsymbol{P}_4\boldsymbol{P}_{i,3}\boldsymbol{P}_5$ 分别是门电路 i 的两个输入电路和输出电线的 RAT.$\boldsymbol{Z}_{i,1},\boldsymbol{Z}_{i,2}$ 被用于随机化门电路 i,上述过程并没有将 RAT 泄露给服务器.重随机化前后的门电路如图 1 所示,重随机化多个门电路如图 2 所示.这时,门电路 1 的输出电线是门电路 2 的输入电线之一,它们具有一致的重随机化状态和同样的 RAT.

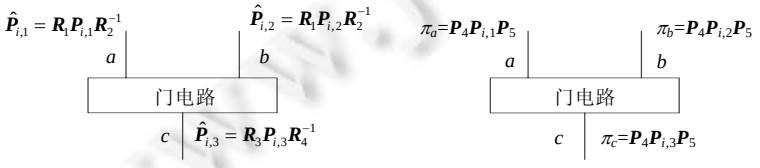

Fig.1 A gate that will be re-randomized and the re-randomized gate

图 1 重随机化前的门电路和重随机化后的门电路

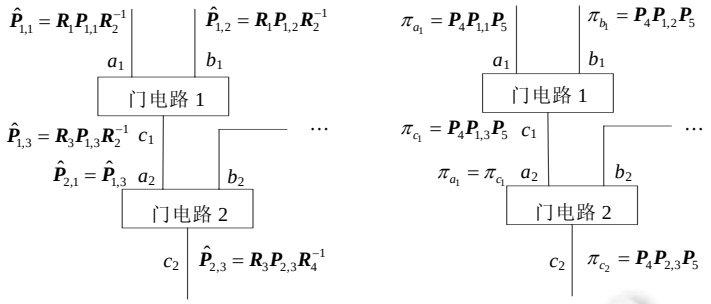

Fig.2 Two gates that will be re-randomized and the re-randomized gates

图 2 重随机化多个门电路前后状态

给定电路输入电线的 IAT(从 RAT $\boldsymbol{P}_4\boldsymbol{P}_{i,1}\boldsymbol{P}_5$ 和 $\boldsymbol{P}_4\boldsymbol{P}_{i,2}\boldsymbol{P}_5$ 易得),客户端的问题生成算法 **ProbGen** 利用 BHHO 加密算法的密钥同态重随机化输入标签.根据这些重随机化标签和重随机化电路,服务器利用 **Compute** 算法逐门电路检查重随机化电路,得到中间以及电路输出门电路由 IAT 随机化的标签.

输出的正确性要求服务器将正确输出交付给客户端,若根本什么都没有交付,则服务器被认为是欺骗或计算错误.Gennaro 等人指出,如果通过检查重随机化混淆电路恢复出正确的电路输出电线标签,则足够表明服务器的电路重随机化过程是诚实的(可参考第 2.3 节)[3].另外,我们的方案还能容忍服务器发起任意次数的验证查询.也就是说,服务器能够获知客户端是否接受或拒绝计算结果.在验证查询下安全的原因是,客户端接受或拒绝决定的比特只与混淆电路的重随机化过程计算相关.

4 可验证计算协议

4.1 协议形式化描述

高层次上的协议描述如下所述.

- 在离线阶段,客户端将函数的混淆电路形式和所有门电路的矩阵 $\hat{\boldsymbol{P}}_{i,1},\hat{\boldsymbol{P}}_{i,2},\hat{\boldsymbol{P}}_{i,3}$ (Kilian 的随机化技术隐藏的映射转换固定部分)发送给服务器.
- 在每次外包计算的在线阶段,客户端将 $\hat{\boldsymbol{P}}_4,\hat{\boldsymbol{P}}_5,\hat{\boldsymbol{P}}_{5,4}$ (Kilian 的随机化技术隐藏的映射转换在每次外包计算中变化部分)和电路输入电线的重随机化标签发送给服务器.

接下来,服务器根据 $\hat{\boldsymbol{P}}_{i,1},\hat{\boldsymbol{P}}_{i,2},\hat{\boldsymbol{P}}_{i,3},\hat{\boldsymbol{P}}_4,\hat{\boldsymbol{P}}_5,\hat{\boldsymbol{P}}_{5,4}$ 计算出 $\boldsymbol{Z}_{i,1},\boldsymbol{Z}_{i,2}$,并重随机化混淆电路.服务器利用客户端的重随机化标签检查重随机化电路,将输出结果返回给客户端,客户端将结果还原出与原混淆电路相对应的标签.

可验证计算协议构造如下所示.

- ***KeyGen***$(1^\lambda, f)\to(pk,sk)$. 将 f 表示为电路 C.该算法由客户端执行如下步骤.
  - 首先,运行混淆电路生成算法产生电路 C 的混淆电路,混淆电路电线 i 标签记为 $L_i^0,L_i^1\in\{0,1\}^\ell$. 具体来说,执行 $Gb.Garble(1^\lambda,C)\to(\Gamma,\{L_i^0,L_i^1\}_{i=1}^{2n})$, 其中,$\Gamma$为混淆电路,$\{L_i^0,L_i^1\}_{i=1}^{2n}$ 是电路输入电线标签,$\{L_i^0,L_i^1\}_{i=n+1}^{2n}$ 是电路输出电线标签.对于门电路 $i,j\in[|\Gamma|]$,随机选择可逆矩阵 $\boldsymbol{P}_{i,1},\boldsymbol{P}_{i,2},\boldsymbol{P}_{i,3}\in\{0,1\}^{(\ell+1)\times(\ell+1)}$ 和 4 个门电路随机掩码$\delta_{i,1},\delta_{i,2},\delta_{i,3},\delta_{i,4}$,为混淆电路也随机选择可逆矩阵:
$$\boldsymbol{R}_1,\boldsymbol{R}_2,\boldsymbol{R}_3,\boldsymbol{R}_4\in\{0,1\}^{(\ell+1)\times(\ell+1)}.$$
  - 接下来,计算 $\hat{\boldsymbol{P}}_{i,1}=\boldsymbol{R}_1\boldsymbol{P}_{i,1}\boldsymbol{R}_2^{-1},\hat{\boldsymbol{P}}_{i,2}=\boldsymbol{R}_1\boldsymbol{P}_{i,2}\boldsymbol{R}_2^{-1}$ 和 $\hat{\boldsymbol{P}}_{i,3}=\boldsymbol{R}_3\boldsymbol{P}_{i,3}\boldsymbol{R}_4^{-1}$. 输出 $pk=(\Gamma,\{\hat{\boldsymbol{P}}_{i,1},\hat{\boldsymbol{P}}_{i,2},\hat{\boldsymbol{P}}_{i,3}\}_{i=1}^{i=|\Gamma|})$ 和
$$sk=(\{L_i^0,L_i^1\}_{i=1}^{2n},\{\boldsymbol{P}_{i,1},\boldsymbol{P}_{i,2},\boldsymbol{P}_{i,3},\delta_{i,1},\delta_{i,2},\delta_{i,3},\delta_{i,4}\}_{i=1}^{i=|\Gamma|},\boldsymbol{R}_1,\boldsymbol{R}_2,\boldsymbol{R}_3).$$
- ***ProbGen***$(sk,x)\to(\sigma_x,\tau_x)$.定义输入 x 为 n 比特大小,即 $x=x_1,\ldots,x_n$.为混淆电路随机选择可逆矩阵 $\boldsymbol{P}_4,\boldsymbol{P}_5\in$

$\{0,1\}^{(\ell+1)\times(\ell+1)}$, 构造矩阵 $\hat{\boldsymbol{P}}_4=\boldsymbol{R}_0\boldsymbol{P}_4\boldsymbol{R}_1^{-1},\hat{\boldsymbol{P}}_5=\boldsymbol{R}_4\boldsymbol{P}_5\boldsymbol{R}_0^{-1},\hat{\boldsymbol{P}}_{5,4}=\boldsymbol{R}_2\boldsymbol{P}_5\boldsymbol{P}_4\boldsymbol{R}_3^{-1}$, 其中,$\boldsymbol{R}_0$ 是大小合适的单位矩阵.执行 $\boldsymbol{Gb.Enc}(\{L_i^0,L_i^1\}_{i=1}^n,(x_1,...,x_n))\rightarrow(c_1,...,c_n)$, 生成输入编码 c.对输入标签 $i\in[n]$,计算:

$$(\gamma_i(c_i)^\top\,|1)^\top=(\boldsymbol{P}_4\boldsymbol{P}_{i,b}\boldsymbol{P}_5)^{-1}.(c_i^T\,|1)^\top,$$

其中,$b\in\{1,2\}$,$\gamma_i(c_i)$表示重随机化输入标签.电路输出电线的 IAT $\eta_1,\ldots,\eta_n$ 也可通过类似计算得到.输出 $\sigma_x=(\gamma_i(c_1),...,\gamma_i(c_n),\hat{\boldsymbol{P}}_4,\hat{\boldsymbol{P}}_5,\hat{\boldsymbol{P}}_{5,4})$ 和 $\tau_x=(\eta_1,\ldots,\eta_n,\boldsymbol{R}_0,\boldsymbol{P}_4,\boldsymbol{P}_5)$.

- $\boldsymbol{Compute}(pk,\sigma_x)\rightarrow(\sigma_y)$.解析 σ_x,对每个 $i\in[|\Gamma|]$,计算 $\boldsymbol{Z}_{i,1}=\hat{\boldsymbol{P}}_4\hat{\boldsymbol{P}}_{i,1}\hat{\boldsymbol{P}}_{5,4}\hat{\boldsymbol{P}}_{i,3}\hat{\boldsymbol{P}}_5,\boldsymbol{Z}_{i,2}=\hat{\boldsymbol{P}}_4\hat{\boldsymbol{P}}_{i,2}\hat{\boldsymbol{P}}_{5,4}\hat{\boldsymbol{P}}_{i,3}\hat{\boldsymbol{P}}_5$,并选择 4 个随机掩码 $\delta'_{i,1},\delta'_{i,2},\delta'_{i,3},\delta'_{i,4}$. 运行 $reRandGb.Gate\{g_i,\boldsymbol{Z}_{i,1},\boldsymbol{Z}_{i,2},(\delta'_{i,1},\delta'_{i,2},\delta'_{i,3},\delta'_{i,4})\}\rightarrow\tilde{g}_i$. 计算 $Gb.Eval\{(\tilde{g}_1,...,\tilde{g}_{|\Gamma|}),(\gamma_1(c_1),...,\gamma_n(c_n))\}\rightarrow(e_1,...,e_n)$.输出 $\sigma_y=(e_1,\ldots,e_n)$.
- $\boldsymbol{Verify}(sk,\sigma_y,\tau_x)\rightarrow(acc,y)$.解析 σ_y 为 $e_1,\ldots,e_n$.如果对输出标签 $i\in[n]$,等式 $L_i=\boldsymbol{reRandGb.OutLabel}(\eta_i,e_i)$成立,则 $acc=1$(接收);否则,$acc=0$(拒绝).如果 $acc=1$,则利用密钥 sk 将 e_i 映射为输出 y;否则,输出$\perp$.

协议的正确性可由混淆电路、可重随机化的混淆电路,数学伪装方法和 Kilian 的随机化技术的正确性直接得出.BHHO 加密算法确保了输入和输出隐私,而随机矩阵链乘使得映射转换也是隐私的,对服务器隐藏的原混淆电路可实现输入和输出隐私的电路重用(见定理 2).协议的离线阶段(执行 **KeyGen**)代价为 $O(poly(\lambda)|C|)$,与函数 f 相关而与被委托输入无关,其中,$|C|$是函数 f 电路的大小.每次外包计算中,客户端外包计算在线的代价是 $O(poly(\lambda)\cdot n)$(执行 **ProbGen**),计算 3 个矩阵链乘代价是 $O(1)$,执行 **Verify** 的代价是 $O(poly(\lambda)\cdot n)$,因为在线代价与函数 f 的电路大小无关,所以该方案是非平凡的(non-trivial).也就是说,客户端自己计算函数 f 的开支比在线代价要大.服务器在线阶段的代价是 $O(poly(\lambda)|C|)$(执行 **Compute**).

文献[3]中具有预见性的方案虽然类似于上述可验证计算协议,但是需要强调与之不同的几点.

- Gennaro 等人给出的可验证计算方案只在敌手不能对客户端发起任何数量的验证查询这种较弱的模型下被证明是安全的.我们的方案能够容忍多项式次数的恶意验证查询.
- Gennaro 等人组合使用 Yao 的混淆电路和 FHE 实现安全的多项式次数输入的混淆电路重用,输入比特相应的标签使用 FHE 加密.为实现多项式次数输入的混淆电路重用,我们使用 BHHO 加密算法的加同态性质重随机化标签和混淆电路.
- Gennaro 等人仅考虑客户端将任意函数计算外包给不被信任的服务器这种非交互可验证计算模式,我们不仅考虑可验证计算,而且也考虑了两方计算下的私有函数计算(private function evaluation,简称 PFE)协议(详见第 5 节).
- 我们的方案是基于 DDH 假设,比 Gennaro 等的方案速度更快、更加紧凑.

4.2 安全分析

利用数学伪装方法可阻止服务器使用已用过的 RAT 重随机化混淆电路,但不包括那些与电路输入电线和电路输出电线相关的 RAT.举例来说,设门电路 i 的 RAT 分别是 $\boldsymbol{P}_{i,1},\boldsymbol{P}_{i,2},\boldsymbol{P}_{i,3}$(不采用数学伪装方法,因此,此时 RAT 是一个矩阵,而不再是 3 个矩阵的乘法链),在每次外包计算中,客户端构造 $\hat{\boldsymbol{P}}_{i,1}=\boldsymbol{R}_0\boldsymbol{P}_{i,1}\boldsymbol{R}_1^{-1},\hat{\boldsymbol{P}}_{i,2}=\boldsymbol{R}_0\boldsymbol{P}_{i,2}\boldsymbol{R}_1^{-1}$ 和 $\hat{\boldsymbol{P}}_{i,3}=\boldsymbol{R}_1\boldsymbol{P}_{i,3}\boldsymbol{R}_0^{-1}$ 并发送给服务器,其中,$\boldsymbol{R}_0$ 是单位矩阵,$\boldsymbol{R}_1$ 是随机置换矩阵(注意,这里仅使用 Kilian 的随机化技术).此时,服务器就可计算 $\boldsymbol{Z}_{i,1}=\hat{\boldsymbol{P}}_{i,1}\hat{\boldsymbol{P}}'_{i,3}$ 和 $\boldsymbol{Z}_{i,2}=\hat{\boldsymbol{P}}_{i,2}\hat{\boldsymbol{P}}'_{i,3}$,其中,$\hat{\boldsymbol{P}}'_{i,3}$ 表示已用于之前外包计算矩阵.另一方面,此时客户端开销与电路的大小具有相同的阶,这样,客户端可仅发送一个新的电路给服务器,实现上这样更容易.

同样地,利用 Kilian 的随机化技术随机隐藏 RAT 的每个部分,限制敌手以基本元素的方式操纵密文组件,比如不按序计算矩阵乘积.敌手有两类可能的攻击 Kilian 的随机化技术方法[25].

- 一类攻击方法是混合输入攻击,敌手正确计算矩阵链乘,但是不遵循矩阵链乘的结构.简而言之,$\hat{\boldsymbol{P}}_{i,1}$ 和 $\hat{\boldsymbol{P}}_{i,2}$ 都前乘和后乘相同的矩阵 $\boldsymbol{R}_1,\boldsymbol{R}_2^{-1}$,服务器可用 $\hat{\boldsymbol{P}}_{i,2}$ 代替 $\hat{\boldsymbol{P}}_{i,1}$ 计算矩阵链乘 $\boldsymbol{Z}_{i,1}$,或者用 $\hat{\boldsymbol{P}}_{i,1}$ 代替 $\hat{\boldsymbol{P}}_{i,2}$ 计算矩阵链乘 $\boldsymbol{Z}_{i,2}$.但是,给定电路输入电线的重随机化标签,服务器在逐个门电路检查重随机化电路时,并不能得到电路输出电线的正确重随化标签[13].

- 另一类攻击方法是部分计算攻击,敌手计算客户端不同输入下的部分矩阵链乘,试图通过比较这些中间值了解 RAT 的一些信息.例如,服务器在 2 个不同客户端输入下分别计算矩阵链乘 $\boldsymbol{Z}_{i,1}$ 的前两个矩阵 $\hat{\boldsymbol{P}}_4\hat{\boldsymbol{P}}_{i,1}$ 乘积,也就是 $\boldsymbol{P}_4\boldsymbol{P}_{i,1}\boldsymbol{R}_2^{-1}$,如果上述 Kilian 矩阵编码方案是理想的,则使用部分计算攻击的敌手并不能获得任何有用信息.

定理 1. 若 DDH 假设是存在的,则上述非交互可验证计算协议对外包函数 f 是安全的.

证明:接下来,采用模拟证明技术(simulation proof technique)[13,26]证明定理 1 成立.首先,先给出引理 1 和引理 2 及其证明.

引理 1. 如果函数 f 是多项式时间可计算函数,由 $Garble(1^\lambda,C)$ 生成混淆电路的分布和由模拟器执行 $GarbleSim(1^\lambda,C)$ 生成混淆电路的分布在 DDH 假设下是计算上不可区分的.

证明:引理 1 的证明过程类似于 Lindell-Pinkas 关于 Yao 协议的证明过程[13].

首先描述模拟器的构造.模拟器执行 $GarbleSim(1^\lambda,C)$,生成一个伪造的混淆电路,过程如下所述:对于混淆电路的每个门电路 g,设其输入电线分别是 a 和 b,输出电线为 c;为电线 a 分别选择活动标签(active label)L_a 和不活动标签(inactive label) L'_a,如果标签被用于计算混淆电路则称它是活动标签,同一电线的另一标签就是不活动标签.同样地,为电线 b 分别选择活动标签和不活动标签 L_b, L'_a,为电线 c 选择活动标签和不活动标签 L_c, L'_c. 为该门电路随机选择 4 个新 2ℓ-bit 掩码 $\delta_1,\delta_2,\delta_3,\delta_4$,计算并随机排序如下 4 个密文对:

$$(Enc_{L_a}(\delta_1), Enc_{L_b}((L_c\,|\,0^\ell)\oplus\delta_1)),$$
$$(Enc_{L_a}(\delta_2), Enc_{L'_b}((L_c\,|\,0^\ell)\oplus\delta_2)),$$
$$(Enc_{L'_a}(\delta_3), Enc_{L_b}((L_c\,|\,0^\ell)\oplus\delta_3)),$$
$$(Enc_{L'_a}(\delta_4), Enc_{L'_b}((L_c\,|\,0^\ell)\oplus\delta_4)).$$

其中,所有密文对只加密输出电线的活动标签.上述所有的门电路构成了 $GarbleSim(1^\lambda,C)$ 生成的混淆电路.

为了证明模拟器产生混淆电路的分布与实际执行 $Garble(1^\lambda,C)$ 生成混淆电路的分布是计算上不可区分的,接下来使用标准的混合体论证(hybrid argument)[26],需定义一系列的混合体 $H_0,H_1,...,H_{|\Gamma|}$.

- H_0:此混合体实际执行 $Garble(1^\lambda,C)$ 生成混淆电路 Γ.
- H_i,其中,$i\in(0,|\Gamma|)$:此混合体与 H_0 的区别在于前 i 个门电路 $g_1,...,g_i$ 的 4 个密文对是由门电路输入标签所有 4 个组合加密门电路输出电线活动标签的密文组成,其他门电路与 H_0 的混淆电路门电路相同.
- $H_{|\Gamma|}$:此混合体执行 $GarbleSim(1^\lambda,C)$ 生成混淆电路,每个门电路都只有输出电线的活动标签加密.

对于所有 $i\in[1,|\Gamma|]$,任意两个连续的混合体 H_{i-1} 和 H_i 的区别在于:H_{i-1} 中门电路 g_i 输出电线不活动标签的密文在 H_i 中被输出电线活动标签的密文所取代.

假设门电路 g_i 的输入电线分别是 a 和 b,输出电线为 c.电线分别是 a 活动标签和不活标签分别是 L_{a_i}, L'_{a_i},相类似电线 b 的分别是 L_{b_i}, L'_{b_i},电线 c 的分别是 L_{c_i}, L'_{c_i}. H_{i-1} 中该门电路的 4 个密文对如下:

$$(Enc_{L_{a_i}}(\delta_{i,1}), Enc_{L_{b_i}}((L_{c_i}\,|\,0^\ell)\oplus\delta_{i,1})),$$
$$(Enc_{L_{a_i}}(\delta_{i,2}), Enc_{L'_{b_i}}((L_{i,1}\,|\,0^\ell)\oplus\delta_{i,2})),$$
$$(Enc_{L'_{a_i}}(\delta_{i,3}), Enc_{L_{b_i}}((L_{i,2}\,|\,0^\ell)\oplus\delta_{i,3})),$$
$$(Enc_{L'_{a_i}}(\delta_{i,4}), Enc_{L'_{b_i}}((L_{i,3}\,|\,0^\ell)\oplus\delta_{i,4})).$$

其中,$L_{i,1},L_{i,2},L_{i,3}$ 是 L_{c_i} 或者是 L'_{c_i}. H_i 中该门电路的 4 个密文对如下:

$$(Enc_{L_{a_i}}(\delta_{i,1}), Enc_{L_{b_i}}((L_{c_i}\,|\,0^\ell)\oplus\delta_{i,1})),$$
$$(Enc_{L_{a_i}}(\delta_{i,2}), Enc_{L'_{b_i}}((L_{c_i}\,|\,0^\ell)\oplus\delta_{i,2})),$$
$$(Enc_{L'_{a_i}}(\delta_{i,3}), Enc_{L_{b_i}}((L_{c_i}\,|\,0^\ell)\oplus\delta_{i,3})),$$
$$(Enc_{L'_{a_i}}(\delta_{i,4}), Enc_{L'_{b_i}}((L_{c_i}\,|\,0^\ell)\oplus\delta_{i,4})).$$

H_{i-1}和H_i是计算上不可区分的,这可通过归约到 BHHO 加密算法的语义安全得出.

假设存在 PPT 区分器 D 和多项式 $p(\cdot)$,对无限多的 n 有:

$$|\Pr[D(H_i)=1]-\Pr[D(H_{i-1})=1]|>\frac{1}{|\Gamma|\,p(n)}.$$

利用区分器 D 构造 PPT 敌手$\mathcal{A}$,$\mathcal{A}$接收门电路 g_i 密文对并构造混淆电路,该电路一部分是 $Garble(1^\lambda,C)$ 真实产生的门电路,另一部分是 $GarbleSim(1^\lambda,C)$ 伪造产生的门电路.若$\mathcal{A}$接收的门电路 g_i 密文对与 H_{i-1} 中的门电路 g_i 密文对相同,则该构造与 H_{i-1} 的混淆电路一致;若$\mathcal{A}$接收的门电路 g_i 密文对与 H_i 中的门电路 g_i 密文对相同,则该构造与 H_i 的混淆电路一致.如果敌手$\mathcal{A}$能够区分 H_{i-1} 和 H_i,就可以区分其中门电路 g_i 的密文对,这与 BHHO 加密算法的安全性相矛盾. □

引理 2. 有效算法 *reRandGb* 输入为随机数 r 和 $Garble'(1^\lambda,C)$ 生成的混淆电路Γ',输出电路 C 的重随机化混淆电路 $\tilde{\Gamma}$,则 $Garble(1^\lambda,C)$ 生成混淆电路Γ的分布和重随机化混淆电路 $\tilde{\Gamma}$ 的分布在 DDH 假设下是计算上不可区分的.

证明:引理 2 的证明方法与引理 1 的类似.为了证明以上两个分布是不可区分的,定义一系列的混合体 H_0,H_1,...,$H_{|T|}$,这里的 T 表示混淆电路的电线数量.

- H_0.此混合体执行 $Gb.Garble(1^\lambda,C)$ 生成混淆电路Γ,使其输出与执行 $Gb.Garble'(1^\lambda,C)$ 生成的混淆电路Γ'输出相同,则混淆电路Γ的分布与混淆电路Γ'的分布是一致的.
- H_i,其中,$i\in(0,|T|)$.此混合体生成的混淆电路前 i 根电线是由重随机化混淆电路Γ'的前 i 根电线得到,其他电线与混淆电路Γ'的电线相同.
- $H_{|T|}$.此混合体是执行 *reRandGb* 生成的重随机化混淆电路 $\tilde{\Gamma}$,每根电线都是重随机化混淆电路Γ'的相应电线而得到的.

对于所有 $i\in[1,|T|]$,任意两个连续的混合体 H_{i-1} 和 H_i 的区别在于第 i 根电线在 H_{i-1} 中与混淆电路Γ'的第 i 根电线相同,而 H_i 的第 i 根电线是由重随机混淆电路Γ'的第 i 根电线而得到的.H_{i-1} 和 H_i 是计算上不可区分的,这可由 BHHO 加密算法安全性得出. □

接下来,利用引理 1 和引理 2 证明定理 1 在两方计算下是成立的,那么定理 1 在可验证计算下也是成立的(我们考虑的不仅是可验证计算,还有在两方计算下的私有函数计算,见第 5 节).基于模拟的安全强于基于不可区分的安全,如果采用模拟证明技术证明协议是基于模拟的安全,则一定是计算不可区分安全(见定义 2).因此,在证明中需构造恶意的服务器和恶意的客户端,并且模拟服务器的视图(view)与客户端的视图.定义一个模拟器 $Sim=\{Sim_c,Sim_s\}$,Sim_c 模拟客户端的视图,Sim_s 模拟服务器的视图.

- Sim_c.给定客户端的输入 x 和计算结果 $y=f(x)$,构造模拟客户端的 Sim_c.首先,一致随机选择矩阵 $\{\hat{\boldsymbol{P}}_{i,1},\hat{\boldsymbol{P}}_{i,2},\hat{\boldsymbol{P}}_{i,3}\}_{i=1}^{i=|\Gamma|}$,$\boldsymbol{A}$,$\boldsymbol{B}$;接下来计算矩阵乘积 $\{\boldsymbol{Z}_{i,1}=\hat{\boldsymbol{P}}_{i,1}\cdot\hat{\boldsymbol{P}}_{i,3},\boldsymbol{Z}_{i,2}=\hat{\boldsymbol{P}}_{i,2}\cdot\hat{\boldsymbol{P}}_{i,3}\}_{i=1}^{i=|\Gamma|}$,为了生成混淆电路$\Gamma$,执行 $Gb.GarbleSim(1^\lambda,C)\rightarrow(\Gamma,\{L_i^0,L_i^1\}_{i=1}^{2n})$;然后选择客户端输入 x 相对应的输入电线活动标签 $\{c_i\}_{i=1}^n$;对输入电线 $i\in[n]$,执行 $\boldsymbol{reRandGb}.\boldsymbol{InLabelSim}(\gamma_i,c_i)\rightarrow\gamma_i(c_i)$,其中,$\gamma_i=A\cdot\hat{\boldsymbol{P}}_{i,1}$ 或 $\gamma_i=B\cdot\hat{\boldsymbol{P}}_{i,2}$. Sim_c 剩下的步骤与客户端执行过程相同.
- Sim_s.给定客户端输入 x 和计算结果 $y=f(x)$,Sim_s 模拟服务器构造混淆电路,其计算结果等于 $f(x)$.具体来说,执行 $Gb.GarbleSim(1^\lambda,C)\rightarrow(\Gamma,\{L_i^0,L_i^1\}_{i=1}^{2n})$, 在 $\{L_i^0,L_i^1\}_{i=1}^n$ 中活动标签与 $\{\gamma_i(c_i)\}_{i=1}^n$ 相关联.考虑输入电线是 a,b 和输出电线为 c 的门电路,可用如等式(1)的 4 个密文对表示.然而,此时 4 个密文对只包含门电路输出电线的活动标签密文.Sim_s 剩下的步骤与服务器执行过程相同.

为了证明协议执行过程的模拟器视图与协议实际执行是计算上不可区分的,需定义一系列的混合体,它们开始于客户端与服务器协议的真实执行,结束于充当客户端与服务器角色的模拟器理想执行.

- H_0.此混合体中的客户端和服务器都遵循协议的实际执行(见第 4.1 节).
- H_1.此混合体与 H_0 的区别仅在于 $\hat{\boldsymbol{P}}_{i,1},\hat{\boldsymbol{P}}_{i,2},\hat{\boldsymbol{P}}_{i,3}$ 的计算方法.模拟器在计算中不采用 Kilian 的随机化技术,

而是为客户端和服务器端都选择随机矩阵 $\hat{\boldsymbol{P}}_{i,1},\hat{\boldsymbol{P}}_{i,2},\hat{\boldsymbol{P}}_{i,3}$.

- H_2.此混合体与 H_1 的区别仅在于 $\boldsymbol{Z}_{i,1},\boldsymbol{Z}_{i,2}$ 的计算方法.Sim_c 在计算中不采用数学伪装方法,而是直接计算 $\boldsymbol{Z}_{i,1}=\hat{\boldsymbol{P}}_{i,1}\cdot\hat{\boldsymbol{P}}_{i,3},\boldsymbol{Z}_{i,2}=\hat{\boldsymbol{P}}_{i,2}\cdot\hat{\boldsymbol{P}}_{i,3}$. 再者,$Sim_s$ 将 $\boldsymbol{Z}_{i,1},\boldsymbol{Z}_{i,2}$ 作为输入,执行 $reRandGb.GateSim\{g_i,\boldsymbol{Z}_{i,1},\boldsymbol{Z}_{i,2},(\delta'_{i,1},\delta'_{i,2},\delta'_{i,3},\delta'_{i,4})\}$, 生成重随机化混淆电路.
- H_3.此混合体与 H_2 的区别仅在于由模拟器新构建混淆电路取代重随机化的混淆电路.具体来说,模拟器模拟客户端执行 $Gb.Garble(1^\lambda,C)$,生成混淆电路;接下来为客户端输入 x 选取电路输入电线标签,执行 $\boldsymbol{Gb.Enc}(gsk,x)$,生成输入的编码.相类似地,模拟器模拟服务器执行 $Gb.Garble(1^\lambda,C)$, 生成混淆电路.
- H_4.此混合体与 H_3 的区别仅在于由模拟器执行 $Gb.GarbleSim(1^\lambda,C)$, 新构建混淆电路取代原有电路.具体来说,模拟器模拟客户端执行 $Gb.GarbleSim(1^\lambda,C)$, 生成混淆电路;接下来为客户端输入 x 选取电路输入电线标签,执行 $\boldsymbol{reRandGb.InLabelSim}(\gamma_i,c_i)$,重随机化输入.相类似地,模拟器模拟服务器执行 $Gb.GarbleSim(1^\lambda,C)$,生成混淆电路.

下面证明每个混合体与相邻的混合体是不可区分的.

引理 3. 对所有的概率多项式时间敌手$\mathcal{A}$, $H_0\overset{C}{\equiv}H_1$.

证明:根据 Kilian 的随机化技术正确性,返回给敌手$\mathcal{A}$的矩阵 $\hat{\boldsymbol{P}}_{i,1},\hat{\boldsymbol{P}}_{i,2},\hat{\boldsymbol{P}}_{i,3}$ 分布上并没有变化.因此,敌手$\mathcal{A}$赢得 H_1 的概率仍然与赢得 H_0 的概率相同. □

引理 4. 对所有的概率多项式时间敌手$\mathcal{A}$, $H_1\overset{C}{\equiv}H_2$.

证明:敌手$\mathcal{A}$赢得 H_1 的概率与赢得 H_2 的概率相同,因为否则就存在一个攻击者对数学伪装方法具有不可忽略的优势. □

引理 5. 对所有的概率多项式时间敌手$\mathcal{A}$, $H_2\overset{C}{\equiv}H_3$.

证明:根据引理 2 的证明可以得出:给定一个由 $Gb.Garble(1^\lambda,C)$ 生成的混淆电路以及由 $reRandGb$ 生成的重随机化混淆电路,没有计算能力受限的敌手能够区分这两个电路.这意味敌手$\mathcal{A}$在 H_3 中的概率与在 H_2 中的概率相同. □

引理 6. 对所有的概率多项式时间敌手$\mathcal{A}$, $H_3\overset{C}{\equiv}H_4$.

证明:根据引理 1 的证明可得出:由 $Gb.GarbleSim(1^\lambda,C)$ 得到的电路与由 $Gb.Garble(1^\lambda,C)$ 得到的电路,没有计算能力受限的敌手能够区分这两个电路.这意味敌手$\mathcal{A}$在 H_4 中的概率与在 H_3 中的概率相同. □

接下来说明在基于模拟的安全下客户端不会接受敌手$\mathcal{A}$伪造的计算结果.若敌手$\mathcal{A}$不能使验证算法接受一个不正确的输出,则可验证计算方案是安全的(见定义 2 关于可验证计算方案是计算不可区分安全的定义).为了说明客户端不会接受敌手$\mathcal{A}$伪造的计算结果,需要给定输入 x 和计算结果 $y=f(x)$,构造客户端模拟器 Sim'_c, 具体过程如下.

1. Sim'_c 生成 $(pk,sk)\leftarrow\boldsymbol{KeyGen}(1^\lambda,f)$.
2. 敌手$\mathcal{A}$向 Sim'_c 发起 **ProbGen** 查询, Sim'_c 执行 $\boldsymbol{ProbGen}(sk,x)$,并将$\sigma_x$返回敌手$\mathcal{A}$.
3. 敌手$\mathcal{A}$向预言机 **PVerify** 发起查询,预言机返回 acc,当且仅当 $\boldsymbol{Verify}(sk,\sigma_x,\tau_x)\rightarrow(acc,y)$.预言机 **Pverify** 仅返回接收/拒绝比特 acc.
4. 步骤 2 和步骤 3 可重复多项式次数.
5. 给定敌手$\mathcal{A}$的σ_y, Sim'_c 生成$(acc,y)\leftarrow\boldsymbol{Verify}(sk,\sigma_y,\tau_x)$.如果$\sigma_y$ 是伪造的计算结果, Sim'_c 将不能映射出正确的混淆电路输出标签,则 acc=0(拒绝),输出$\perp$;否则,acc=1(接收),并输出 y.

Sim'_c 的输出与在真实协议中客户端的输出消息是不可区分的,这可从引理 1~引理 6 的证明过程得出.

综上所述,定理 1 得证. □

定理 2. 若 DDH 假设是存在的,则上述非交互可验证计算协议对服务器是隐私的.

证明:该证明与定理 1 的证明都具有类似形式的混合体和证明过程.服务器的隐私性可由 RAT 的隐私、可重随机化的混淆电路安全性和 Yao 的混淆电路安全性得出. □

4.3 适应性安全

静态安全的混淆电路是指输入不依赖于混淆电路[13,27](见定义 7).与之相反,适应性安全的混淆电路是指,如果敌手在查看混淆电路以后还允许适应性地选择输入,此时混淆电路仍是安全的(见定义 8).Yao 的混淆电路只能做到静态安全的,这是因为为了提高在线阶段的效率,混淆电路的生成和发送通常都在离线阶段,恶意的敌手就有可能根据混淆电路自己选择输入,使得混淆电路不再安全.文献[3]的可验证计算方案使用了 Yao 的混淆电路,为了保证方案的安全,必须在发送混淆电路之前确定输入,因此该方案是静态安全的.

第 4.1 节的可验证计算协议虽然也是静态安全的,但可采用两种方法实现适应性安全:一种方法是将该协议运用复杂性杠杆(complexity leveraging)实现适应性安全;另一种方法是将该协议作细微的调整,即可做到适应性安全.为了使恶意的敌手在离线阶段不能根据混淆电路自己选择输入,客户端只需将所有门电路的密文对的前半部分和后半部分别与一个大小合适的随机可逆矩阵 $\boldsymbol{R}_5^{-1}$ 相乘.相应的,构造矩阵 $\hat{\boldsymbol{P}}_4=\boldsymbol{R}_0\boldsymbol{P}_4\boldsymbol{R}_1^{-1}$ 修改为 $\hat{\boldsymbol{P}}_4=\boldsymbol{R}_5\boldsymbol{P}_4\boldsymbol{R}_1^{-1}$,计算 $\boldsymbol{Z}_{i,1}=\boldsymbol{R}_5\boldsymbol{P}_4\boldsymbol{P}_{i,1}\boldsymbol{P}_5\boldsymbol{P}_4\boldsymbol{P}_{i,3}\boldsymbol{P}_5$ 和 $\boldsymbol{Z}_{i,2}=\boldsymbol{R}_5\boldsymbol{P}_4\boldsymbol{P}_{i,2}\boldsymbol{P}_5\boldsymbol{P}_4\boldsymbol{P}_{i,3}\boldsymbol{P}_5$,再将现在的 $\boldsymbol{Z}_{i,1},\boldsymbol{Z}_{i,2}$ 用于随机化门电路 i.上述过程既没有将 RAT 泄露给敌手,也保证敌手不能尝试混淆电路的输入,即实现了适应性安全.为了说明这一点,可采用文献[27]中静态安全转换为适应性安全的构造方法.具体来说,因为定理 1 成立,故存在静态安全 PPT 模拟器 $\mathcal{S}$,对任意 PPT 敌手 $\mathcal{A}$(见定义 7),

$$\mathrm{pr}[\boldsymbol{Exp}_{\mathcal{A},s}^{static}(1^{\lambda},0)=1]-\mathrm{pr}[\boldsymbol{Exp}_{\mathcal{A},s}^{static}(1^{\lambda},1)=1]\leqslant negl(\lambda).$$

给定任意适应性安全 PPT 敌手 $\mathcal{A}_1$,构建静态安全敌手 $\mathcal{A}$.若由模拟器 $\mathcal{S}$ 能够构建适应性安全 PPT 模拟器 $\mathcal{S}_1$:

$$\mathrm{pr}[\boldsymbol{Exp}_{\mathcal{A}_1,\mathcal{S}_1}^{adaptive}(1^{\lambda},0)=1]-\mathrm{pr}[\boldsymbol{Exp}_{\mathcal{A}_1,\mathcal{S}_1}^{adaptive}(1^{\lambda},1)=1]\leqslant negl(\lambda) \tag{3}$$

则适应性安全成立(见定义 8).

适应性安全实验 $\boldsymbol{Exp}_{\mathcal{A}_1,\mathcal{S}_1}^{adaptive}(1^{\lambda},b_1)$ 定义如下.

1. 挑战者随机选择比特 b_1.
2. 敌手 $\mathcal{A}_1$ 向挑战者提交 C,挑战者返回 $\tilde{\Gamma}$.若 $b_1=0$,挑战者返回随机混淆电路 $\tilde{\Gamma}$;如果 $b_1=1$,挑战者也返回随机混淆电路 $\tilde{\Gamma}$.
3. 敌手 $\mathcal{A}_1$ 向挑战者提交 x.
4. 如果 $b_1=0$,则挑战者调用定义 7 的步骤 3,生成 $(\tilde{\Gamma}',gsk)\leftarrow \boldsymbol{Gb}.\boldsymbol{Garble}(1^{\lambda},C)$ 和 $\tilde{c}\leftarrow \boldsymbol{Gb}.\boldsymbol{Enc}(gsk,x)$,并将 $\tilde{\Gamma}'$ 所有门电路的密文对的前半部分和后半部分别与 $\boldsymbol{R}_5^{-1}$ 相乘生成 $\tilde{\Gamma}$,并返回 $\tilde{\Gamma}$ 和 $\tilde{c}$.
5. 如果 $b_1=1$,则挑战者调用定义 7 的步骤 4,生成 $(\tilde{\Gamma}'',state)\leftarrow S(1^{\lambda},T(C))$ 和 $\tilde{c}\leftarrow S(C(x),state)$,并将 $\tilde{\Gamma}''$ 所有门电路的密文对的前半部分和后半部分别与 $\boldsymbol{R}_5^{-1}$ 相乘生成 $\tilde{\Gamma}$,并返回 $\tilde{\Gamma}$ 和 $\tilde{c}$.
6. 最后,敌手 $\mathcal{A}_1$ 输出猜测比特 b_1',如果 $b_1'=b_1$,敌手 $\mathcal{A}_1$ 获胜.

设 b 是定义 7 的挑战比特,从上述适应性安全实验可以得出:

$$pr[\boldsymbol{Exp}_{\mathcal{A},s}^{static}(1^{\lambda},1)=1]=pr[\boldsymbol{Exp}_{\mathcal{A}_1,\mathcal{S}_1}^{adaptive}(1^{\lambda},1)=1],$$

$$pr[\boldsymbol{Exp}_{\mathcal{A},s}^{static}(1^{\lambda},0)=1]=pr[\boldsymbol{Exp}_{\mathcal{A}_1,\mathcal{S}_1}^{adaptive}(1^{\lambda},0)=1].$$

所以,等式(3)成立.

5 可重用的密码转置防火墙

接下来,将第 4.1 节的协议应用于密码转置防火墙[23],从而提供一种密码转置防火墙的新模式,即可重用的密码转置防火墙.密码转置防火墙聚焦于用户与外部通讯的密码安全,可解释为一个状态算法,应用于用户发送和接收已由某些密码算法处理的消息.两方计算下的私有函数计算 PFE 协议有一对参与方分别是 Alic 和 Bob,Alice 拥有私有的函数 f,Bob 拥有私有的输入 x,双方计算 $f(x)$并保证输入 x 和函数 f 的隐私性和结果的正确性.

PFE 协议的密码转置防火墙主要技术工具是可重随机化的不经意传输和可重随机化的混淆电路.Bob(Bob 的密码转置防火墙)和 Alice(Alice 的密码转置防火墙)执行可重随机化的不经意传输,Alice 向 Bob 透露电路输入电线的两个重随机化标签其中之一,而不了解确切是哪一个.接下来,Alice 的密码转置防火墙将重随机化的混淆电路发送给 Bob,Bob 就可以根据重随机化标签计算电路.然而,Alice 的密码转置防火墙重用用于重随机化电路的 Yao 的混淆电路是不安全的,这是因为重复的重随机化等同于混淆电路的重用.所以,我们提出一种可重用的密码转置防火墙,用户一次生成混淆电路,接下来,密码转置防火墙可安全地重随机化多次.

5.1 不经意传输

Naor-Pinks/Aiello-Ishai-Reingold 证明了不经意传输协议可在诚实但好奇模式下是安全的[28,29].不经意传输协议有两个参与方:发送方 Alice 和接收方 Bob.Alice 的输入为 $a_0,a_1\in\{0,1\}$;Bob 的输入是选择的比特 $b\in\{0,1\}$.Alice 和 Bob 之间协议实现如图 3 所示,其中,$\mathbb{G}$是阶为 q 的群.

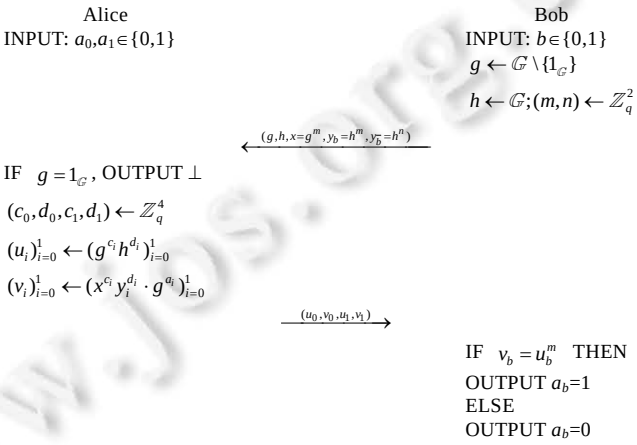

Fig.3 Oblivious transfer (OT) protocol

图 3 不经意传输协议 OT

定义 10(可重随机化的不经意传输). 如图 3 所示两消息的不经意传输协议,设 $msg_1=(g,h,x,y_0,y_1)$是第 1 轮的消息,$msg_2=(u_0,v_0,u_1,v_1)$是第 2 轮的消息.

不经意传输协议是可重随机化的,若存在输入为 msg_1,msg_2 和随机数 r 的有效算法 reRandOT,即使给定 r,b,a_0,a_1,msg_1,msg_2,以下两个分布是计算上不可区分的.

$$reRandOT(msg_1,msg_2,r)\overset{C}{\equiv}(u_0,v_0,u_1,v_1).$$

5.2 私有函数计算的可重用密码转置防火墙

Alice 的 PFE 可重用密码转置防火墙 W_A 的构造与可验证计算协议的构造非常类似,技术上的区别在于,W_A 需要定义如何构造不经意传输的重随机化.因为 Bob 的可重用密码转置防火墙 W_B 与文献[23]的不经意传输协议重随机化是一致的,故此处省略.构造 Alice 的 PFE 可重用转置密码防火墙如图 4 所示.本质上该构造与可验证计算协议相同,安全要求相同,证明也类似.

定理 3. 若 DDH 假设成立,如图 4 所示的 Alice 的可重用密码转置防火墙对于函数 f 是正确的和安全的.

证明:Alice 的可重用密码转置防火墙的正确性证明如下.

$\mathbb{G}$是素数阶 q 的循环群,定义 $g\leftarrow\mathbb{G}\setminus\{1_{\mathbb{G}}\},k=\log_g^h.(g,h,x,y_0,y_1)$是 Bob 发送给 Alice 的初始消息,Bob 的防火墙安全性可直接由 DDH 假设得出.恶意模式下的可重随机化不经意传输安全和可重随机化混淆电路安全确保 Alice 的防火墙是安全的.接下来证明可重随机化不经意传输的安全性.

Fig.4 Alice's reusable cryptographic rerverse firewall for the private function evaluation protocol

图 4 私有函数计算协议中 Alice 的可重用密码转置防火墙

W_B·(W_A·*Alice*)对来自 Bob 的任意有效消息(g,h,x,y_0,y_1)以不可忽略的概率做出一致随机有效响应 (u_0',v_0',u_1',v_1'). 若 $(g,h,x,y_b,y_{\bar{b}}) \neq (g,g^k,g^m,g^{km},g^{kn})$, 则 (u_0',v_0',u_1',v_1') 是一致随机群元素.为了说明这一点,需要利用等式 $\log_g^{u_i} = c_i + kd_i$ 和 $\log_g^{v_i/g^{a_i}} = mc_i + kmd_i$ (或者 $\log_g^{v_i/g^{a_i}} = mc_i + knd_i$).若 $y_b \neq g^{km}$(或 $y_b \neq g^{kn}$), (u_0',v_0',u_1',v_1') 的分布是一致和独立的.

Alice 的可重用密码转置防火墙安全意味已知 $(g,h,x,y_b,y_{\bar{b}})$ 可以模拟 (u_0',v_0',u_1',v_1').为了证明这一点,需要给定 $(g,h,x,y_b,y_{\bar{b}})$ 和有效计算结果 a_b,构造模拟器 Sim_A.模拟器的操作如下.

1. 输入比特σ,Sim_A 产生 $(g,h,x,y_\sigma,y_{\bar{\sigma}})$.
2. 随机选择$(a_0,a_1) \leftarrow \{0,1\}$和 $(c_0,c_1,d_0,d_1) \leftarrow \mathbb{Z}_q^4$.
3. 计算 $(u_i,v_i)_{i=0}^1 \leftarrow \{g^{c_i}h^{d_i}, x^{c_i}y_i^{d_i} \cdot g^{a_i}\}_{i=0}^1$.
4. 同样地,选择 $(c_0',c_1',d_0',d_1') \leftarrow \mathbb{Z}_q^4$.
5. 同样地,计算 $(u_i',v_i')_{i=0}^1 \leftarrow \{g^{c_i'}h^{d_i'} \cdot u_i, x^{c_i'}y_i^{d_i'} \cdot v_i\}_{i=0}^1$.
6. 输出 $(\sigma,u_0',v_0',u_1',v_1')$.

Sim_A 的输出与在真实协议中 Bob 接收来自 Alice 的可重用密码防火墙的消息是不可区分的,这可从 DDH 假设得出. □

References:

[1] Kilian J. A note on efficient zero-knowledge proofs and arguments (extended abstract). In: Proc. of the 24th Annual ACM Symp. on Theory of Computing. ACM Press, 1992. 723−732.

[2] Micali S. CS proofs (extended abstract). In: Proc. of the 35th Annual Symp. on Foundations of Computer Science. IEEE Press, 1994. 436−453.

[3] Gennaro R, Gentry C, Parno B. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Rabin T, ed. Proc. of the 30th Annual Cryptology Conf. (CRYPTO 2010). Heidelberg: Springer-Verlag, 2010. 465−482.

[4] Applebaum B, Ishai Y, Kushilevitz E. From secrecy to soundness: efficient verification via secure computation. In: Abramsky S, *et al*., eds. Proc. of the 37th Int'l Colloquium on Automata, Languages, and Programming (ICALP 2010). Heidelberg: Springer-Verlag, 2010. 152−163.

[5] Asharov G, Jain A, Lopez-Alt A, Tromer E, Vaikuntanathan V, Wichs D. Multiparty computation with low communication, computation and interaction via threshold FHE. In: Pointcheval D, Johansson T, eds. Proc. of the 31st Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2012). Heidelberg: Springer-Verlag, 2012. 483−501.

[6] Ben-Sasson E, Chiesa A, Genkin D, Tromer E, Virza M. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Canetti R, Garay JA, eds. Proc. of the 33rd Annual Cryptology Conf. (CRYPTO 2013). Heidelberg: Springer-Verlag, 2013. 99−108.

[7] Choi SG, Katz J, Kumaresan R, Cid C. Multi-client non-interactive verifiable computation. In Sahai A, ed. Proc. of the 10th Theory of Cryptography Conf. (TCC 2013). Heidelberg: Springer-Verlag, 2013. 499−518.

[8] Chung KM, Kalai YT, Liu FH, Raz R. Memory delegation. In Rogaway P, ed. Proc. of the 31st Annual Cryptology Conf. (CRYPTO 2011). Heidelberg: Springer-Verlag, 2011. 151−168.

[9] Chung KM, Kalai Y, Vadhan S. Improved delegation of computation using fully homomorphic encryption. In: Rabin T, ed. Proc. of the 30th Annual Cryptology Conf. (CRYPTO 2010). Heidelberg: Springer-Verlag, 2010. 483−501.

[10] Fiore D, Gennaro R, Pastro V. Efficiently verifiable computation on encrypted data. In: Proc. of the 21st ACM Conf. on Computer and Communications Security (CCS 2014). ACM Press, 2014. 844−855.

[11] Ananth P, Chandran N, Goyal V, Kanukurthi B, Ostrovsky R. Achieving privacy in verifiable computation with multiple servers—Without FHE and without pre-processing. In: Krawczyk H, ed. Proc. of the 20th Int'l Conf. on Practice and Theory of Public-Key Cryptography (PKC 2014). Heidelberg: Springer-Verlag, 2014. 149−166.

[12] Badrinarayanan S, Goyal V, Jain A, Sahai A. Verifiable functional encryption. In: Cheon J, Takagi T, eds. Proc. of the 22nd Annual Int'l Conf. on the Theory and Applications of Cryptology and Information Security (ASIACRYPT 2016). Heidelberg: Springer-Verlag, 2016. 557−587.

[13] Lindell Y, Pinkas B. A proof of security of Yao's protocol for two-party computation. Journal of Cryptology, 2009,22(2):161−188.

[14] Yao AC. Protocols for secure computations (extended abstract). In: Shamir A, ed. Proc. of the 23rd Annual Symp. on Foundations of Computer Science. IEEE Press, 1982. 160−164.

[15] Goldwasser S, Kalai Y, Popa RA, Vaikuntanathan V, Zeldovich N. Reusable garbled circuits and succinct functional encryption. In: Proc. of the 45th Annual ACM Symp. on Theory of Computing (STOC 2013). ACM Press, 2013. 555−564.

[16] Agrawal S. Stronger security for reusable garbled circuits, general definitions and attacks. In: Katz J, Shacham H, eds. Proc. of the 37th Annual Cryptology Conf. (CRYPTO 2017). Heidelberg: Springer-Verlag, 2017. 3−35.

[17] Gentry C. Fully homomorphic encryption using ideal lattices. In: Proc. of the 41st Annual ACM Symp. on Theory of Computing (STOC 2009). ACM Press, 2009. 169−178.

[18] Brakerski Z, Vaikuntanathan V. Lattice-based FHE as secure as PKE. In: Proc. of the 5th Conf. on Innovations in Theoretical Computer Science. ACM Press, 2014. 1−12.

[19] Gentry C, Halevi S, Vaikuntanathan V. I-hop homomorphic encryption and re-randomizable Yao circuits. In: Rabin T, ed. Proc. of the 30th Annual Cryptology Conf. (CRYPTO 2010). Heidelberg: Springer-Verlag, 2010. 155−172.

[20] Halevi S, Lindell Y, Pinkas B. Secure computation on the Web: Computing without simultaneous interaction. In: Rogaway P, ed. Proc. of the 31st Annual Cryptology Conf. (CRYPTO 2011). Heidelberg: Springer-Verlag, 2011. 132−150.

[21] Atallah MJ, Pantazopoulos KN, Rice JR. Secure outsourcing of scientific computations. Advances in Computers, 2002,(54): 215−272.

[22] Kilian J. Founding cryptography on oblivious transfer. In: Proc. of the 20th Annual ACM Symp. on Theory of Computing. ACM Press, 1988. 20−31.

[23] Mironov I, Stephens-Davidowitz N. Cryptographic reverse firewalls. In: Oswald E, Fischlin M, eds. Proc. of the 34th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2015). Heidelberg: Springer-Verlag, 2015. 657−686.

[24] Boneh D, Halevi S, Hamburg M, Ostrovsky R. Circular-secure encryption from decision Diffie-Hellman. In: Wagner D, ed. Proc. of the 28th Annual Cryptology Conf. (CRYPTO 2008). Heidelberg: Springer-Verlag, 2008. 108−125.

[25] Barak B, Garg S, Kalai YT, Paneth O, Sahai A. Protecting obfuscation against algebraic attacks. In: Nguyen PQ, Oswald E, eds. Proc. of the 33rd Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2014). Heidelberg: Springer-Verlag, 2014. 221−238.

[26] Lindell Y. How to simulate it—A tutorial on the simulation proof technique. IACR Cryptology ePrint Archive: Report 2016/046 (2016), 2016. http://eprint.iacr.org/2016/046.pdf

[27] Bellare M, Hoang VT, Rogaway P. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In: Wang X, Sako K, eds. Proc. of the 18th Int'l Conf. on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2012). Heidelberg: Springer-Verlag, 2012. 134−153.

[28] Aiello W, Ishai Y, Reingold O. Priced oblivious transfer: How to sell digital goods. In: Pfitzmann B, ed. Proc. of the 20th Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2011). Heidelberg: Springer-Verlag, 2001. 119−135.

[29] Naor M, Pinkas B. Efficient oblivious transfer protocols. In: Proc. of the 12th Annual ACM-SIAM Symp. on Discrete Algorithms. ACM Press, 2001. 448−457.

赵青松(1973－),男,江苏连云港人,博士,讲师,CCF 专业会员,主要研究领域为信息安全和隐私,公钥密码学.

刘西蒙(1988－),男,博士,教授,CCF 专业会员,主要研究领域为应用密码学,数据安全,安全计算,公钥密码学.

曾庆凯(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为信息安全,分布计算.

徐焕良(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为分布计算,数据科学与计算.