计算机科学2003Vol. 30№. 8

安全多方计算协议的研究与应用*)

李 强 颜 浩 陈克非
(上海交通大学计算机科学与工程系 上海200030)

Research and Application of Secure Multi-party Computation Protocols

LI Qiang YAN Hao CHEN Ke-Fei
(Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200030)

Abstract This paper introduces the current research status of the multi-party computation protocols, briefly shows the current four types of multi-party computation protocols and summarizes the research direction.

Keywords Multi-party computing

1 安全多方计算协议简介

安全多方计算协议要解决的问题可以描述如下:一组参与者希望共同计算某个约定的函数,每个参与者提供函数的一个输入,出于安全考虑,要求参与者提供的输入对其他人保密。如果存在安全可信第三方(我们称其为 PPT),则安全多方协议所要解决的问题可以轻易地得到解决:只需各参与者将各自的输入交给 PPT,由 PPT 计算出函数值,再将函数值公布给各参与者。但现实中很难找到这样的 PPT,从而安全多方计算协议的研究应运而生,目前安全多方计算已得到许多学者的研究,其在密码学上的地位也日益重要,它是电子选举、电子拍卖等密码学协议的基础。A. C. Yao 于1982年最早提出了两方安全计算协议[1]。之后,O. Goldreich, S. Micali 及 A. Wigderson 提出了可以计算任意函数的基于密码学安全模型的安全多方计算协议[2],证明了存在被动攻击者时 n-Secure 的协议是存在的;存在主动攻击时(n−1)-Secure 的协议是存在的;D. Chaum, C. Crepeau 和 I. Damgard 中对信息论安全模型下的安全多方计算进行了研究,证明了在被动攻击下(n−1)-Secure 的协议是存在的,在主动攻击下($\lfloor n/2 \rfloor -1$)-seure 的协议是存在的[3]。此后,许多学者在如何提高安全多方计算协议的效率,如何对安全多方计算进行形式化的定义,如何对通用的安全多方计算协议进行剪裁使之能更有效地适用于不同的应用环境,新的安全多方计算协议的构造方法,安全多方计算攻击者结构定义等方面进行了大量的研究。S. Goldwasser 和 L. Levin 对计算模型在大部分协议参与者(超过半数)都被买通情况下的安全多方计算协议进行了研究[4]。O. Goldreich, S. Goldwasser 和 N. Linial 对不安全的通讯信道、拥有无限计算能力的攻击者模型下的安全多方计算协议进行了研究[5]。R. Ostrovsky 和 M. Yung 在安全信道模型下对移动攻击者(Mobile Adversaries)进行了研究[6]。S. Micali, P. Rogaway[7]和 Beaver[8]对安全信道模型下的安全多方计算给出了形式化的定义。

2 常用安全多方计算协议

下面简要介绍一下目前常用的4类安全多方计算协议。

2.1 基于 OT 的安全多方计算协议

这种安全多方计算协议的基础是 OT 子协议:发送者 S 有 k 个输入 $d_1, d_2, \cdots, d_k \in \{0,1\}$,接收者 R 有输入 $l \in \{1,2,\cdots,k\}$,协议执行后,R 得到 d_l 而不知道除 d_l 外的 S 的其他输入,S 不知道 R 的选择 l。为表述方便,我们将 OT 子协议记为 $OT[(d_1,d_2,\cdots,d_k),l,S,R]$。OT 子协议有许多构造方法,利用任何公钥密码系统都可以构造 OT 子协议。

下面介绍如何利用 OT 子协议进行安全多方计算。此类安全多方计算可以计算任意的比特运算函数,我们知道所有的比特运算都可以分解成二元 AND、XOR 运算及一元 NOT 运算的组合,因此只要利用 OT 子协议可以安全地计算二元 AND、XOR 运算及一元 NOT 运算,则可以安全地计算所有比特运算函数(为了表述方便,我们将 XOR 运算记为⊕,将 AND 运算记成·,将 NOT 运算记成−)。其计算过程可以大致地分为3个步骤:

输入阶段:n 个参与者 $P_1, \cdots, P_n$ 拥有各自的函数输入自变量 $x_i (i=1,2,\cdots,n) \in \{0,1\}$,$P_i$ 将其自变量 x_i 随机分解成 $x_{i,1}, x_{i,2}, \cdots, x_{i,n}$ 使得 $x_i = x_{i,1} \oplus x_{i,2} \oplus \cdots \oplus x_{i,n}$,并将 $x_{i,j}$ 秘密发送给 P_j。

计算阶段:

(1)二元 XOR 运算。设运算逻辑上的输入分别为 a, b(a, b 可以是初始输入,也可以是计算过程的中间结果),且 a, b 已经分别被表示成 $a = a_1 \oplus a_2 \oplus \cdots \oplus a_n$,$b = b_1 \oplus b_2 \oplus \cdots \oplus b_n$,其中 P_i 只知道 a_i, b_i。则经过 XOR 运算后,各 P_i 将其 XOR 运算输出设置为 $a_i \oplus b_i$。运算的正确性可由下式保证:

$$a \oplus b = (a_1 \oplus a_2 \oplus \cdots \oplus a_n) \oplus (b_1 \oplus b_2 \oplus \cdots \oplus b_n)$$
$$= (a_1 \oplus b_1) \oplus (a_2 \oplus b_2) \oplus \cdots \oplus (a_n \oplus b_n)$$

(2)一元 NOT 运算。设运算逻辑上的输入为 a(a 可以是初始输入,也可以是计算过程的中间结果),且 a 已经被表示成 $a = a_1 \oplus a_2 \oplus \cdots \oplus a_n$,其中 P_i 只知道 a_i。则经过 NOT 运算后,P_1 将其 NOT 运算输出设置为 $\overline{a_1}$,其他参与者 NOT 运算输出设置成 $a_i (i=2,3,\cdots,n)$。运算的正确性可由下式保证:

$$\overline{a} = \overline{a_1 \oplus a_2 \oplus \cdots \oplus a_n} = \overline{a_1} \oplus a_2 \oplus \cdots \oplus a_n$$

(3)二元 AND 运算。设运算逻辑上的输入分别为 a, b(a, b 可以是初始输入,也可以是计算过程的中间结果),且 a, b 已经分别被表示成 $a = a_1 \oplus a_2 \oplus \cdots \oplus a_n$,$b = b_1 \oplus b_2 \cdots \oplus b_n$,其中 P_i 只知道 a_i, b_i。我们的目标是得到 $c_1, c_2, \cdots, c_n$,保证 P_i 只知

*)本研究得到国家自然科学基金项目69973031, 90104005资助。李 强 硕士研究生。颜 浩 硕士研究生。陈克非 博士,教授,博士生导师。

道 c_i，并满足下式：

$$c_1\oplus c_2\oplus\cdots\oplus c_n=a\cdot b=(a_1\oplus a_2\oplus\cdots\oplus a_n)\cdot(b_1\oplus b_2\oplus\cdots\oplus b_n)$$

先看 $n=2$ 的特殊情况，这时候我们的目标是得到 c_1,c_2，保证 P_1 只知道 c_1，P_2 只知道 c_2，使得 $c_1\oplus c_2=(a_1+a_2)\cdot(b_1\oplus b_2)$。可以利用 $OT[(d_1,d_2,d_3,d_4),l,S,R]$ 达到我们的目标：P_1 充当 S 的角色，P_2 充当 R 的角色，P_1 随机地选择 $c_1\in\{0,1\}$，并设置各参数如下 $d_1=c_1\oplus(a_1\cdot b_1)$，$d_2=c_1\oplus[a_1\cdot(b_1\oplus 1)]$，$d_3=c_1\oplus[(a_1\oplus 1)\cdot b_1]$，$d_4=c_1\oplus[(a_1\oplus 1)\cdot(b_1\oplus 1)]$ 作为 OT 的输入，P_2 设置 $l=1+2a_2+b_2\in\{1,2,3,4\}$ 作为 OT 的输入，执行 $OT[(d_1,d_2,d_3,d_4),l,P_1,P_2]$ 协议，由 OT 的性质，P_2 只能得到 $d_l=d_{1+2a_2+b_2}$ 而 P_1 不知道 $l=1+2a_2+b_2$。P_2 设置 $c_2=d_l=d_{1+2a_2+b_2}$，很容易验证 $c_1\oplus c_2=(a_1+a_2)\cdot(b_1\oplus b)$，且 P_1 只知道 c_1，P_2 只知道 c_2。为了方便描述，我们记 $OT_S(S,R)=c_1$，$OT_R(S,R)=c_2$。

对于一般情况，由于

$$\left(\bigoplus_{i=1}^{n}a_i\right)\cdot\left(\bigoplus_{i=1}^{n}b_i\right)=\bigoplus_{i=1}^{n}(a_i\cdot b_i)\oplus\bigoplus_{1\le i<j\le n}[(a_i\cdot b_j)\oplus(a_j\cdot b_i)]$$
$$=(n-2)\left[\bigoplus_{i=1}^{n}(a_i\cdot b_i)\right]\oplus\bigoplus_{1\le i<j\le n}[(a_i\oplus a_j)\cdot(b_i\oplus b_j)]$$

不难看到，P_i 可以单独计算 $(n-2)(a_i\cdot b_i)=\bigoplus_{j=1}^{n-2}(a_i\cdot b_i)$，任意 $\{P_i,P_j\}(1\le i<j\le n)$ 可以通过执行 $OT[(d_1,d_2,d_3,d_4),l,P_i,P_j]$ 协议，使得 P_i 得到 $OT_{P_i}(P_i,P_j)$，P_j 得到 $OT_{P_j}(P_i,P_j)$，满足：

$$OT_{P_i}(P_i,P_j)\oplus OT_{P_j}(P_i,P_j)=(a_i\oplus a_j)\cdot(b_i\oplus b_j)$$

P_i 设置 $c_i=\bigoplus_{j=1}^{n-2}(a_i\cdot b_i)\oplus\bigoplus_{j=1}^{i-1}OT_{P_i}(P_j,P_i)\oplus\bigoplus_{j=i+1}^{n}OT_{P_i}(P_i,P_j)$，则由上面的推导有 $c_1\oplus c_2\oplus\cdots\oplus c_n=(a_1\oplus a_2\oplus\cdots\oplus a_n)\cdot(b_1\oplus b_2\oplus\cdots\oplus b_n)=a\cdot b$，且 P_i 只知道 c_i 而对 $c_j(j\neq i)$ 没有任何信息。

输出阶段：对于需要安全计算的函数，设经过最后一步运算（二元 AND、XOR 运算或一元 NOT 运算）后，P_i 得到 y_i，很容易看出函数的输出结果为 $y=y_1\oplus y_2\oplus\cdots\oplus y_n$。$P_i$ 将 y_i 公布，则所有 P_i 正确得到了 y。

上面介绍的基于 OT 的安全多方计算协议要求所有的协议参与者正确地执行涉及的每步运算。要处理某些参与者在参与过程中有恶意行为的情况，则在协议中还需用到比特承诺、投币协议、零知识证明等子协议，具体参见文[9]。

2.2 使用 VSS 子协议的安全多方计算协议

这种类型的安全多方计算协议使用了秘密分享协议，由于要求保证被分享的秘密的可验证性，需要使用可验证的秘密分享协议（VSS）。目前这类安全多方计算协议很多，如文[10,11]等，下面介绍一下文[11]中提到的安全多方计算协议。与基于 OT 的安全多方计算协议类似，这类安全多方计算协议分为输入阶段、计算阶段、输出阶段。进行安全多方计算的函数的自变量及函数值的取值域为 $G(p)$，其中 p 为一素数，由于域上的所有函数都可以分解成为域上加法及乘法运算的组合，因此只要能够安全计算域上加法及乘法运算，则可以安全计算域上所有函数：

输入阶段：n 个参与者 $P_1,\cdots,P_n$ 拥有各自的输入自变量 $s_i(i=1,2,\cdots,n)\in G(p)$，$P_i$ 采用随机 t 次多项式 $f_i(x)=a_{i,t}x^t+\cdots+a_{i,2}x^2+a_{i,1}x+s_i$ 将 s_i 分享，并将 $f_i(j)$ 秘密发送给 P_j。

计算阶段：

（1）域上加法运算。设运算逻辑上的输入分别为 $a,b\in G(p)$（a,b 可以是初始输入，也可以是计算过程的中间结果），且 a,b 分别被分享在 $G(p)$ 上 t 次随机多项式 $f_a(x)$ 及 $f_b(x)$ 中，即 $f_a(0)=a$，$f_b(0)=b$，其中 P_i 只知道 $f_a(i)$，$f_b(i)$。则经过加法运算后，各 P_i 将其加法运算输出设置为 $f_a(i)+f_b(i)$，也就是说 $a+b$ 被分享在 $G(p)$ 上 t 次随机多项式 $h(x)=f_a(x)+f_b(x)$ 中，其中 $a+b=f_a(0)+f_b(0)=h(0)$，P_i 只知道 $h(i)=f_a(i)+f_b(i)$。加法运算的正确性是显然的。

（2）域上的乘法运算。设运算逻辑上的输入分别为 $a,b\in G(p)$（a,b 可以是初始输入，也可以是计算过程的中间结果），且 a,b 分别被分享在 $G(p)$ 上 t 次随机多项式 $f_a(x)$ 及 $f_b(x)$ 中，即 $f_a(0)=a$，$f_b(0)=b$，其中 P_i 只知道 $f_a(i)$，$f_b(i)$。则域上的乘法运算的结果要求：找到 $G(p)$ 上 t 次一随机多项式 $h(x)$ 分享 ab（即 $h(0)=ab$），使得 P_i 只知道子秘密 $h(i)$ 而不知道其他子秘密。记 $f_{ab}(x)=f_a(x)f_b(x)=a_{2t}x^{2t}+\cdots+a_2x^2+a_1x+ab$（注意 $f_{ab}(x)$ 是 $2t$ 次多项式，且 $f_{ab}(0)=f_a(0)f_b(0)=ab$），则

$$A\cdot X=\begin{bmatrix}1&1&\cdots&1\\1&2&\cdots&2^{2t}\\\cdots&\cdots&\cdots&\cdots\\1\cdots&2t+1&\cdots&(2t+1)^{2t}\end{bmatrix}\begin{bmatrix}ab\\a_1\\\cdots\\a_{2t}\end{bmatrix}=\begin{bmatrix}f_{ab}(1)\\f_{ab}(2)\\\cdots\\f_{ab}(2t+1)\end{bmatrix}$$

其中，A 为范德蒙矩阵，可逆。设其逆矩阵 A^{-1} 为：

$$A^{-1}=\begin{bmatrix}\lambda_1&\lambda_2&\cdots&\lambda_{2t+1}\\\cdots&\cdots&\cdots&\cdots\\\cdots&\cdots&\cdots&\cdots\\\cdots&\cdots&\cdots&\cdots\end{bmatrix}$$

则 $ab=\lambda_1f_{ab}(1)+\cdots\lambda_{2t+1}f_{ab}(2t+1)$，注意 $\lambda_1,\lambda_2,\cdots,\lambda_{2t+1}$ 对所有人都是已知的，$f_{ab}(i)$ 只对 P_i 已知，从而只要 $P_i(i=1,2,\cdots,2t+1)$ 用 $G(p)$ 上 t 次随机多项式 $h_i(x)$ 分享 $\lambda_if_{ab}(i)$（即 $h_i(0)=\lambda_if_{ab}(i)$）将 $\lambda_if_{ab}(i)$ 分享，并将 $h_i(j)$ 秘密发送给 P_j，令 $h(x)=h_1(x)+h_2(x)+\cdots h_{2t+1}(x)$，则 $h(0)=\sum_{i=1}^{2t+1}h_i(0)=\sum_{i=1}^{2t+1}\lambda_if_{ab}(i)=ab$，$P_i$ 知道 $\sum_{j=1}^{2t+1}h_j(i)=h(i)$ 而对 $h(x)$ 的其他函数值一无所知。从而完成了域上的乘法运算。

输出阶段：对于需要安全计算的函数，设经过最后一步运算（$G(p)$ 域上加法或乘法运算）后，P_i 得到 $y_i=y(i)$，则由 Shamir 秘密共享方案的性质，任意 $t+1$ 可以共同恢复 $y(0)$，容易知道 $y(0)$ 是函数的输出值。

上面介绍的安全多方计算协议要求所有的协议参与者正确地执行涉及的每步运算。要处理某些参与者在参与过程中有恶意行为的情况，则在协议中还需用到同态承诺、检测 VSS 的 VSPS（可验证的多项式秘密分享）性质、零知识证明等子协议，具体参见文[11]。

2.3 基于同态加密的安全多方计算协议

此类安全多方计算协议需使用同态加密技术，同态加密技术是满足如下性质的公钥加密技术：设 $\bar{a}$ 表示明文 a 的加密密文，（1）加法加密同态：如果已知 $\bar{a},\bar{b}$，任何人都能得到 $a+b$ 的密文 $\overline{a+b}$（记 $\overline{a+b}$ 为 $\bar{a}\oplus\bar{b}$）；（2）常数乘法同态：如果已知 $\bar{a}$ 及一常数 a，任何人都能轻易得到 aa 的密文 $\overline{aa}$；（3）已知

明文证明：若某人知道 a，他可以利用零知识证明 $\overline{a}$ 是 a 的密文；(4)常数乘法正确性证明：若某人已知 α 及 $\overline{a}$，他可以使用 $\overline{a}$，$\overline{a}$，$\overline{\alpha a}$作为输入，零知识证明$\overline{\alpha a}$确实是 αa 的密文；(5)门限解密：用于加密的公钥 pk 公开，用于解密的私钥被分享在参与者中，解密 $\overline{a}$ 时，各参与者使用其子私钥作为输入(每次解密不需要公开子私钥)，可以共同解密出 a。如文[12,13]就是这样的同态加密系统。这类安全多方计算协议同样包括输入阶段、计算阶段、输入阶段：

输入阶段：n 个参与者 $P_1,\cdots,P_n$ 拥有各自的输入自变量 x_i，P_i 使用 pk 将 x_i 加密，将密文$\overline{x_i}$发送给每个协议的参与者，并零知识证明他知道 x_i。

计算阶段：

(1)加法运算。每个协议的参与者都知道加法运算的输入 a,b 对应的密文 $\overline{a},\overline{b}$，由加法加密同态性质，每个人都能得到 $\overline{a}\oplus\overline{b}=\overline{a+b}$。

(2)乘法运算。每个协议的参与者都知道乘法运算的输入 a,b 对应的密文 $\overline{a},\overline{b}$，目标是每个参与者能够得到$\overline{ab}$：$P_i$ 随机选择 d_i 并使用 pk 将其加密，将$\overline{d_i}$广播并证明他知道 d_i；由加法加密同态性质，每个参与者都知道$\overline{d_1}\oplus\overline{d_2}\oplus\cdots\oplus\overline{d_n}\oplus\overline{a}=\overline{d_1+d_2+\cdots+d_n+a}$，所有协议参与者共同解密得到 $d_1+d_2+\cdots+d_n+a$(注意，对每个参与者而言，这只是一个随机数)；由于 $d_i,a+\sum_{j=1}^{n}d_j$ 对 P_i 而言是已知的，P 设置 $a_1=(a+\sum_{j=1}^{n}d_j)-d_1$，其他 P_i 设置 $a_i=-d_i(i=2,3,\cdots,n)$，显然 $\sum_{i=1}^{n}a_i=a$；由常数乘法同态性质，P_i 可以计算$\overline{a_ib}$，P_i 公布$\overline{a_ib}$并使用常数乘法正确性证明其运算是真实的；由加法加密同态性质，每个 P_i 知道$\overline{a_jb}(j=1,2,\cdots,n)$，从而知道$\overline{a_1b}\oplus\overline{a_2b}\oplus\cdots\oplus\overline{a_nb}=\overline{a_1b+a_2b+\cdots+a_nb}=\overline{ab}$。

输出阶段：对于需要安全计算的函数，设经过最后一步运算(加法或乘法运算)后得到的密文为 $\overline{y}$，所有参与者共同解密出 y 就是整个函数的输出。

上面的描述没有介绍在已知明文证明或常数乘法证明验证出错、门限解密时有参与者恶意破坏时协议如何处理，这种情况的处理详见文[14]。

2.4 使用 Mix_Match 的安全多方计算协议

这是一种使用盲表进行安全多方计算的协议，适用于逻辑运算和比特运算，可以在电子拍卖等协议中使用。与前面介绍的3种安全多方计算协议不同，它使用 Mix_Match 提供协议所需的保密性和正确性。这种安全多方计算使用了 Elgamal 加密方案的同态性质及门限解密性质。首先简要介绍一下 Elgamal 加密方案：p 是大素数，g 是 $G(p)$的生成元，私钥 $k\in G(p)$对应的公钥为 $y=g^k$，使用 y 对 $m\in G(p)$加密的密文为$(\alpha,\beta)=(my^r,g^r)$，其中 $r\in G(p)$为随机数，解密过程为 $m=\alpha/\beta^k$。Elgamal 加密方案的同态性质：对于相同的系统 g，p 参数及公私钥对 k,y，设(α_0,β_0)是 m_0的密文，(α_1,β_1)是 m_1的密文，则$(\alpha_0\alpha_1,\beta_0\beta_1)$是 m_0m_1的密文；进一步设(γ,η)是1的密文，则$(\alpha_0\gamma,\beta_0\eta)$是 m_0的另一个密文。Elgamal 加密方案的门限解密性质：设私钥 k 使用 $t-1$次多项式 $f(x)=a_{t-1}x^{t-1}+\cdots+a_2x^2+a_1x+k$ 在参与者中分享，P_i 得到 $k_i=f(i)$，要解密(α,β)，P_i 计算并公布 $\beta_i=\beta^{k_i}$，选择 t 个 $\beta_{s_1},\beta_{s_2},\cdots,\beta_{s_t}$，则 $m=\alpha/(\prod_{i=1}^{t}\beta_{s_i}^{\lambda_{s_i}})$，其中 $\lambda_{s_1},\lambda_{s_2},\cdots,\lambda_{s_t}$满足 $k=\lambda_{s_1}\beta_{s_1}+\lambda_{s_2}\beta_{s_2}+\cdots+\lambda_{s_t}\beta_{s_t}$。这种类型的安全多方计算可以分为4个阶段：系统建立阶段、输入阶段、计算阶段、输出阶段。

系统建立阶段：

(1)Elgamal 系统分布式建立：所有协议参与者约定一个大素数 p 及 $G(p)$的生成元 g，每个 P_i 选择一个随机数 r_i 并使用一随机 $t-1$次多项式 $f_i(x)=a_{i,t-1}x^{t-1}+\cdots+a_{i,2}x^2+a_{i,1}x+r_i$，公布 $y_i=g^{r_i}$，将 $f_i(j)$秘密发送给 P_j；当 P_i 收到所有其他参与者发送的 $f_j(i)$后，计算 $y=\prod_{j=1}^{n}y_j$，$k_i=\sum_{j=1}^{n}f_j(i)$，容易验证系统分布式建立了私钥 $k=\sum_{j=1}^{n}r_j$，并将私钥以 $t-1$次随机多项式 $f(x)=\sum_{j=1}^{n}f_j(x)$分享在所有参与者中，$P_i$ 得到 $k_i=f(i)$，k 对应的公钥为 y。

(2)混合网络(Mix_Network)建立：所有协议参与者的一部分 $P_{c_1},P_{c_2},\cdots,P_{c_m}$组成建立混合网络的服务器。将计算中可能涉及的所有门电路(注意是比特运算)的真值表进行混合运算：设门电路的真值表共有 l 行 h 列，表示成：

$$T=\begin{bmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,h} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,h} \\ \cdots & \cdots & \cdots & \cdots \\ t_{l,1} & t_{l,2} & \cdots & t_{l,h} \end{bmatrix}$$

其中 $t_{i,j}\in\{-1,1\}(t_{i,j}=-1$表示逻辑上的0)。$P_{c_1}$使用公钥 y 将表 T 中的所有元素进行随机加密，然后将 T 的各行进行随机置换(注意只是行置换，没有列置换)，得到表 T_1；P_{c_2}使用公钥 y 将 T_1的所有元素进行再次加密(利用加密系统的同态性质，将 $t_{i,j}$的密文与1的随机加密结果组合成 $t_{i,j}$的另一个密文)，并将 T_1的各行进行随机置换得到 T_2；类似地，$P_{c_{i+1}}$使用公钥 y 将 T_i 的所有元素进行再次加密、对各行进行随机置换得到 T_{i+1}；最后输出的 T_m 为门电路真值表 T 的随机密文表(各行元素已被随机加密，各行的顺序已随机打乱)。

输入阶段：n 个参与者，每个参与者 P_i 拥有计算函数的某个输入 x_i(若 $x_i=0$则以 -1代替)，使用公钥 y 将 x_i 随机加密，将加密结果$\overline{x_i}$公布。

计算阶段：所有参与者共同执行 PET(明文相等测试)子协议进行查表运算：当遇到要计算的门电路时，将门电路输入值的密文(初始输入或中间结果)与该门电路真值表随机密文表的每一行(最后一个元素除外)进行对照，检测是否都表示相同明文的密文，如果查到与某行一致，则将该行最后一个元素(表示该门电路对应于输入的函数值的密文)作为查表结果，即该门电路运算的结果。其中检测两密文是否表示相同明文要使用 PET 子协议：设(α_0,β_0)，(α_1,β_1)分别表示 m_0,m_1的密文，如果 $m_0=m_1$，则$(\gamma,\eta)=(\alpha_0/\alpha_1,\beta_0/\beta_1)$表示1的密文，则$(\gamma^z,\eta^z)$($z$ 为某随机数)仍是1的密文，否则(γ^z,η^z)表示$(m_0/m_1)^z$为一随机数(因为 z 为随机数)；各 P_i 随机选择 z_i 计算并公布$(\gamma_i,\eta_i)=(\gamma^{z_i},\eta^{z_i})$，令 $\tilde{\gamma}=\prod_{i=1}^{n}\gamma_i$，$\tilde{\eta}=\prod_{i=1}^{n}\eta_i$，所有参与者共同参与解密$(\tilde{\gamma},\tilde{\eta})$，如果解密结果为1则表示$(\alpha_0,\beta_0)$，$(\alpha_1,\beta_1)$表示相同明文对应的密文，否则表示不是。

输出阶段：执行函数的最后一个门电路后，设输出的密文为(α,β)，所有参与者共同参与解密(α,β)，得到的结果即为函数的运算结果。

与前面介绍的几种类型的安全多方计算一样，上面介绍的安全多方计算没有考虑参与者恶意破坏的情况，这种情况

的处理详见文[15]。

3 安全多方计算协议的研究方向

概括起来，目前众多的研究者的研究主要集中在如下一些方面：(1)一般化的安全多方计算协议：这类研究的目的是设计一种高效的、安全的、能够计算任意函数的安全多方计算协议，希望能够通过这样一种协议一劳永逸地解决所有的涉及到安全多方计算的问题。(2)对一般化的安全多方计算协议进行剪裁：这类研究注意到如果一个协议是广泛适用的，那么必然会牺牲一些性能上的代价来满足其广泛使用性。针对于某一个具体的应用，如电子选举，对一般化的安全多方计算协议进行一点剪裁，去掉一些对某个具体应用没有意义的部分，可以大大地提高效率。这类研究主要研究的是如何将安全多方计算的理论和技术应用到一个具体的应用中去。(3)安全多方计算的定义：安全多方计算的目的和应该具备的性质都为在这一领域研究的学者熟知了，但是安全多方计算至今还缺乏一个为大家所认同的、完备的定义。这主要是因为安全多方计算协议的构造形式可以有多种，各个学者研究的安全模型也是不一样的。(4)新的安全多方计算协议的构造方法：目前大部分学者提出的安全多方计算协议都使用了 VSS 子协议作为其构造基石，因而这类协议在大体结构上都是类似的。有没有其他的方法来构造安全多方计算协议?这也是一些学者的研究内容。(5)安全多方计算协议生成器：许多安全多方计算协议的研究在研究了如何安全地计算域上定义的基本运算就停止了，因为这时从理论上讲，任何函数都可以被安全地计算。但是这离实际应用还有一步需要跨越，多方计算协议生成器的作用就是弥补这最后的一步。安全多方计算协议生成器的输入是任意函数和如何计算域上定义的基本运算的子协议，输出是安全计算该函数的协议。许多学者提到了对方计算协议生成器，但是没有对之进行细致的研究。

参考文献

1 Yao A C. Protocols for secure computations. In:Proc. 23rd IEEE Symp. On the Foundation of Computer Science, IEEE, 1982. 160～164

2 Goldreich O, Micali S, Wigderson A. How to play any mental game. In:Proc. 19th ACM Symp. On the Theory of Computing, 1987. 218～229

3 Chaum D, Crepeau C, Damgard I. Multiparty unconditionally secure protocols (extended abstract). In:Proc. 20th ACM Symp. On the Theory of Computing, 1988. 11～19

4 Goldwasser S, Levin L. Fair computation of general functions in presence of immoral majority. In Advances in Cryptology-CRYPTO'90 volume 537 of LNCS. Springer-Verlag, 1990

5 Goldreich O, Goldwasser S, Linial N. Fault-Tolerant Computation in the Full Information Model. 32nd FOCS, 1991. 447～457

6 Ostrovsky R, Yung M. How to withstand mobile virus attacks. In:Proc. of the 10th Annual ACM Symposium on Principles of Distributed Computing, 1991. 51～59

7 Micali S, Rogaway P. Secure Computation. CRYPTO, 1991

8 Beaver D. Foundations of Secure Interactive Computing. CRYPTO, 1991

9 Goldreich O. Secure multi-party computation(working draft, Version 1.1). 1998

10 Chor B, Goldwasser S, Micali S, Awerbuch B. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. In: Proc. 26th Annual Symposium on the Foundations of Computer Science, IEEE, 1985. 383～395

11 Gennaro R, Rabin M, Rabin T. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In:Proc. of the 1998 ACM Symposium on Principles of Distributed Computing

12 Matthew, Franklin, Habert S. Joint encryption and message-efficient secure computation. Journal of Cryptology, 1996, 9(4):217～232

13 Paillier P. Public-key cryptosystems based on composite degree residue classes. In:Michael Wiener, ed. Advances in Cryptology-EuroCrypt'99, Berlin, 1999. 223～238

14 Cramer R, Damgard I, Nielsen J B. Multiparty Computation form Threshold Homomorphic Encryption. BRICS. June, 2000

15 Jakobsson M, Juels A. Mix and Match:Secure Function Evaluation via Ciphertexts

(上接第30页)

22 Fan Li, Cao Pei, Jacobson Q. Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance. In:Proc. of the ACM SIGMETRICS'99, Atlanta, Georgia, May 1999

23 Douglis F, Feldmann A, Krishnamurthy B, Mogul J. Rate of change and other metrics: a live study of the world wide web. In: Proc. of the USENIX Symposium on Internet Technologies and Systems, Monterey, California, Dec. 1997

24 Feldmann A, C'aceres R, Douglis F, Glass G, Rabinovich M. Performance of Web proxy caching in heterogeneous bandwidth environments. In:Proc. of the IEEE INFOCOM'99 Conf. 1999

25 Kroeger T M, Long D D E, Mogul J C. Exploring the bounds of web latency reduction from caching and prefetching. In:Proc. of the USENIX Symposium on Internet Technologies and Systems, Dec. 1997. 13～22

26 Thompson K, Miller G, Wilder R. Wide-area Internet traffic patterns and characteristics. In: Proc. 3td Int'l. conf. Web Caching, 1998

27 Venkataramani A, et al. The potential costs and benefits of long-term prefetching for contentdistribution. In: Sixth International Workshop on Web Caching and Content Distribution, June 2001

28 Cao P, Zhang J, Beach K. Active cache: caching dynamic contents on the web. In: Proc. IFIP Int'l. Conf. Dist. Sys. Platforms and open Dist. processing, 1998

29 Gwertzman J S. Autonomous replication across wide-area internetworks. Thesis, Harvaed College, Cambridge, MA, 1995

30 Davison B D. Assertion: prefetching with get is not good. In: the Proc. of the 6th Intl. Workshop on Web Caching and Content Distrribution, June. 2001. 20～22

31 Dodge R, Menasce D A. Prefetching inlines to improve web server latency. In: the Proc. of the 1998 Computer Measurement Group Conf. Anaheim, CA, Dec. 1998. 6～11

32 Gitzenis S, Bambos N. Power-controlled data prefetching/caching in wireless packet network. IEEE. 2002

33 Khan J I, Tao Qingping. Partial Prefetch for Faster Surfing in Composite Hypermedia. In: the Proc. of the 3rd USENIX Symposium on Internet Technologies USITS'01San Francisco March 2001. 13～24

34 Jiang Yingyin, Wu Min-You, Shu Wei. Web Prefetching: Costs, Benefits and Performance

35 An-Chow Lai, Cem Fide and Babak Falsafi. In:Proc. Of the 28th Annual International Symposium on Computer Architecture 2002