论 文

可完全验证的双线性对运算外包算法

任艳丽^{①*}, 丁宁^{②③}, 王天银^④, 陆海宁^⑤, 谷大武^②^① 上海大学通信与信息工程学院, 上海 200444^② 上海交通大学电子信息与电气工程学院, 上海 200240^③ 密码科学技术国家重点实验室, 北京 100878^④ 洛阳师范学院数学科学学院, 洛阳 471022^⑤ 上海交通大学信息安全工程学院, 上海 200240

* 通信作者. E-mail: renyanli@shu.edu.cn

收稿日期: 2016-01-22; 接受日期: 2016-02-03

国家自然科学基金 (批准号: 61202367, 61572309, 61572246)、教育部博士点基金 (批准号: 20120073110094)、上海市教委创新基金 (批准号: 14YZ020)、河南省科技创新人才支持计划和河南省高校科技创新人才支持计划 (批准号: 13HASTIT042) 资助项目

摘要 双线性对运算是基于对的密码学中最昂贵的运算. 基于两个不可信的服务器, 本文首先提出了单个双线性对运算的外包计算方案. 用户不需要执行任何复杂度高的运算, 如标量乘、模乘运算等. 本文还提出了多个双线性对运算的外包方案, 效率远远高于调用单个对运算外包方案. 在所提的两个算法中, 如果服务器不诚实, 用户均能以概率 1 检测到错误. 基于两个算法, 本文还提出了可外包解密的基于身份匿名加密方案和签名外包验证的基于身份签名方案.

关键词 云计算 可验证 外包计算 双线性对 模指数运算 基于身份加密

1 引言

随着云计算的发展和移动设备的普及, 外包计算也引起了人们的广泛关注. 将计算任务外包给云服务器可大大降低用户的计算代价, 但由于服务器并不是完全可信的, 因此也遇到许多安全挑战. 首先, 用户数据经常会包含一些隐私信息, 而这些信息用户并不想泄露给云服务器. 第二, 云服务器可能会返回不正确的计算结果, 但是用户并不能检测到这些错误^[1].

可验证计算 (verifiable computation, VC) 允许计算能力受限的用户将计算任务外包给功能强大但不可信的服务器^[2,3]. 用户首先执行大量的离线计算加密计算函数, 然后将加密后的函数外包给服务器. 服务器在加密后的函数上进行计算, 并返回计算结果及正确性的证明. 最终, 用户基于服务器的证明, 验证计算结果是否正确. 在整个过程中, 用户的计算代价远远小于直接计算.

在密码学领域, 将昂贵的计算外包给半可信的设备受到广泛研究. Chaum 等^[4] 引入了 “wallets with observers” 的概念, 在用户主机上安装硬件设备执行复杂运算. Hohenberger 等^[5] 正式定义了这个模型, 并基于两个不能相互勾结的服务器提出了模指数运算的外包算法. Chen 等^[1] 基于相同模型, 提出了模指数运算的改进算法, 同时提高了外包算法的计算效率和可验证概率. Green 等^[6] 为基于属

引用格式: 任艳丽, 丁宁, 王天银, 等. 可完全验证的双线性对运算外包算法. 中国科学: 信息科学, 2016, 46: 855–869, doi: 10.1360/N112016-00020

性加密 (attribute based encryption, ABE) 方案提出了外包解密算法, 降低了用户的解密代价. 但是, 算法不能验证外包结果的正确性. Lai 等^[7] 在 ABE 方案中提出了可验证的外包解密算法, 用户能有效检测外包结果的正确性. 其他外包计算的工作包括单向函数求逆^[8]、矩阵相乘^[9] 等.

双线性对在基于身份加密^[10]、短签名^[11]、密钥协商协议^[12] 等领域有广泛应用. 在基于对的密码学中, 双线性对的计算是最费时的, 对方案或协议的效率有重要影响. 因此, 人们在提高双线性对的计算效率方面做了大量工作^[13~16].

Chevallier-Mames 等^[17] 基于一个不可信的服务器, 提出了双线性对运算的外包方案. 如果服务器不诚实, 用户能以概率 1 检测到错误. 但是, 用户需要执行一些复杂度很高的运算, 如标量乘 (scalar multiplications)、模指数 (modular exponentiations) 运算等, 这些运算的复杂度与双线性对运算的计算代价几乎是等价的^[16, 18]. 随后, 关于双线性对的外包工作都存在这个问题^[19, 20]. Chen 等^[21] 基于两个不可信的服务器, 提出了第一个实用的双线性对运算外包算法, 用户只需要执行 5 次点加和 4 次模乘运算, 不需要任何复杂度高的运算, 适用于计算受限的设备. 但是, 算法的可验证概率只有 1/2, 即服务器还是能以 1/2 的概率欺骗用户. 最近, Tian 等^[22] 基于两个服务器提出了两个双线性运算的外包算法. 一个提高了外包算法的效率, 用户只需执行 4 次点加和 3 次乘法, 可验证概率同样是 1/2. 另一个提高可验证概率至 $(1 - \frac{1}{3s})^2$, 其中 s 是一个小的整数, 因此用户还是有可能被服务器欺骗, 接受错误的计算结果.

本文基于两个不可信的服务器, 提出了单个双线性对运算的外包算法, 提高了用户的可验证概率, 并将所提算法推广到多个双线性对运算. 在提出的两个算法中, 如果服务器不诚实, 用户均能以概率 1 检测到错误. 最终, 使用这两个算法, 提出了可外包解密的基于身份匿名加密方案和签名外包验证的基于身份签名方案.

2 外包算法的定义和安全模型

本节介绍双线性对的性质, 外包算法的定义和安全模型.

2.1 双线性对

令 q 是一个大素数, G 和 $\hat{G}$ 是两个阶为 q 的加法群, G_T 是阶为 q 的乘法群, P 和 $\hat{P}$ 分别是 G 和 $\hat{G}$ 的生成元. 如果满足如下性质, 称 $e: G \times \hat{G} \rightarrow G_T$ 是一个双线性映射^[23~26]:

- (1) 双线性. 对任意 $U \in G, V \in \hat{G}, a_0, b_0 \in \mathbb{Z}_q^*$, 有 $e(a_0U, b_0V) = e(U, V)^{a_0b_0}$.
- (2) 非退化性. 存在 $R \in G, Q \in \hat{G}$, 满足 $e(R, Q) \neq 1$.
- (3) 可计算性. 对任意 $U \in G, V \in \hat{G}$, 存在有效算法计算 $e(U, V)$.

双线性映射和双线性对可通过基于有限群的超奇异椭圆曲线上的 Weil 或 Tate 对实现^[23~26].

2.2 定义和安全模型

本小节介绍外包计算的定义和安全模型^[5]. 算法 Alg 包括可信方 T 和不可信的程序 U , T^U 代表 T 调用 U 进行运算. 敌手 A 被一组算法 (E, U') 模拟, 其中 E 代表为算法 Alg 提交恶意输入的恶意环境, U' 代表 E 撰写的恶意软件. 如文献^[5] 所述, 假定两个敌手 (E, U') 只能在 T^U 执行之前直接通信, 否则只能通过用户 T 通信.

定义 1 (外包定义) 算法 Alg 包括 5 个输入, 生成 3 个输出. 前 3 个输入是可信方 T 选择的, 后两个输入是恶意环境 E 选择的. 第 1 个输入是诚实、保密的, 对 E 和 U' 都是未知的; 第 2 个是诚实、受保护的, 对 E 是公开的, 对 U' 是保密的; 第 3 个是诚实、不受保护的, 对 E 和 U' 都是公开的; 第 4 个是恶意、受保护的, 对 E 是公开的, 对 U' 是保密的; 最后一个为恶意、不受保护的, 对 E 和 U' 都是公开的. 同样地, 第 1 个输出是保密的, 对 E 和 U' 都是未知的; 第 2 个是受保护的, 对 E 是公开的, 对 U' 是保密的; 第 3 个是不受保护的, 对 E 和 U' 都是公开的.

下面的安全定义确保: 即使 T 使用 E 撰写的软件 U' , E 和 U' 也不能获得外包算法 T^U 的输入和输出.

定义 2 (安全模型) Alg 如定义 1 所示. 如果满足以下条件, 称 T^U 是 Alg 的一次安全运行.

(1) 正确性. T^U 是 Alg 的一次正确运行.

(2) 保密性. 对所有概率多项式 (probabilistic polynomial-time, PPT) 敌手 $A = (E, U')$, 存在两个 PPT 模拟器 (S_1, S_2) , 对以下两组随机变量在计算上是不可区分的.

Pair one. $\text{EVIEW}_{\text{real}} \sim \text{EVIEW}_{\text{ideal}}$, 即恶意环境 E 在 T^U 执行过程中不能获得输入和输出的任何信息. Real process 和 ideal process 都是逐轮执行的.

The real process:

$$\begin{aligned} \text{EVIEW}_{\text{real}}^i &= \{(\text{istate}^i, x_{hs}^i, x_{hp}^i, x_{hu}^i) \leftarrow I(1^k, \text{istate}^{i-1})\}; \\ (\text{estate}^i, j^i, x_{ap}^i, x_{au}^i, \text{stop}^i) &\leftarrow E(1^k, \text{EVIEW}_{\text{real}}^{i-1}, x_{hp}^i, x_{hu}^i); \\ (\text{tstate}^i, \text{ustate}^i, y_s^i, y_p^i, y_u^i) &\leftarrow \{T^{U'}(\text{ustate}^{i-1})(\text{tstate}^{i-1}, x_{hs}^i, x_{hp}^i, x_{hu}^i, x_{ap}^i, x_{au}^i) : (\text{estate}^i, y_p^i, y_u^i)\}; \\ \text{EVIEW}_{\text{real}} &= \text{EVIEW}_{\text{real}}^i \text{ if } \text{stop}^i = \text{TRUE}. \end{aligned}$$

在第 i 轮, 诚实的进程 I 选择诚实、秘密的; 诚实、受保护的; 诚实、不受保护的 3 个输入 $(x_{hs}^i, x_{hp}^i, x_{hu}^i)$, 而恶意环境 E 不能获得这些输入. 此后, E 基于上一轮的状态 $\text{EVIEW}_{\text{real}}^{i-1}$ 和 $T^{U'}$ 的输入 $(x_{hs}^i, x_{hp}^i, x_{hu}^i)$, 选择 $\text{estate}^i, j^i, x_{ap}^i, x_{au}^i, \text{stop}^i$, 其中 estate^i 是一个变量, 用来提醒下次调用的操作, x_{ap}^i, x_{au}^i 是两个敌手输入, stop^i 是一个 Boole 函数, 决定第 i 轮是否是最后一轮. 然后, 基于输入 $(\text{tstate}^{i-1}, x_{hs}^i, x_{hp}^i, x_{hu}^i, x_{ap}^i, x_{au}^i)$, 算法 $T^{U'}$ 执行进程, 为 T 生成新的状态 tstate^i , 及秘密的、受保护的、不受保护的输出 y_s^i, y_p^i, y_u^i , 其中 tstate^{i-1} 是 T 上一次保存的状态. U' 基于以前的状态 ustate^{i-1} , 保存现在的状态 ustate^i . 在第 i 轮, the real process 的输出是 estate^i, y_p^i and y_u^i . 在整个 real process, U 的输出是最后一轮的输出, 此时 $\text{stop}^i = \text{TRUE}$.

The ideal process:

$$\begin{aligned} \text{EVIEW}_{\text{ideal}}^i &= \{(\text{istate}^i, x_{hs}^i, x_{hp}^i, x_{hu}^i) \leftarrow I(1^k, \text{istate}^{i-1})\}; \\ (\text{estate}^i, j^i, x_{ap}^i, x_{au}^i, \text{stop}^i) &\leftarrow E(1^k, \text{EVIEW}_{\text{ideal}}^{i-1}, x_{hp}^i, x_{hu}^i); \\ (\text{astate}^i, y_s^i, y_p^i, y_u^i) &\leftarrow \{Alg(\text{astate}^{i-1}, x_{hs}^i, x_{hp}^i, x_{hu}^i, x_{ap}^i, x_{au}^i)\}; \\ (\text{sstate}^i, \text{ustate}^i, Y_p^i, Y_u^i, \text{rep}^i) &\leftarrow S_1^{U'}(\text{ustate}^{i-1})(\text{sstate}^{i-1}, \dots, x_{hp}^i, x_{hu}^i, x_{ap}^i, x_{au}^i, y_p^i, y_u^i); \\ (z_p^i, z_u^i) &= \text{rep}^i(Y_p^i, Y_u^i) + (1 - \text{rep}^i)(y_p^i, y_u^i) : (\text{estate}^i, z_p^i, z_u^i); \\ \text{EVIEW}_{\text{ideal}} &= \text{EVIEW}_{\text{ideal}}^i \text{ if } \text{stop}^i = \text{TRUE}. \end{aligned}$$

在 ideal process 中, 模拟器 S_1 不能得到秘密输入 x_{hs}^i , 但是可以得到第 i 轮算法 Alg 的受保护的和不受保护的输出, 然后根据 Boole 变量 rep^i , 决定输出 (y_p^i, y_u^i) , 还是用其他值 (Y_p^i, Y_u^i) 代替. 在整个过程中, S_1 可以询问 U' , U' 同在 real process 一样保存自己的状态.

Pair two. $\text{UVIEW}_{\text{real}} \sim \text{UVIEW}_{\text{ideal}}$, 即由恶意环境 E 撰写的不可信软件 U' 在 T^U 执行过程中不能获得输入和输出的任何信息.

同 Pair one, 如果 $\text{stop}^i = \text{TRUE}$, U' 在 real process 的状态为 $\text{UVIEW}_{\text{real}} = \text{ustate}^i$.

The ideal process:

$$\begin{aligned} \text{UVIEW}_{\text{ideal}}^i &= \{(\text{istate}^i, x_{hs}^i, x_{hp}^i, x_{hu}^i) \leftarrow I(1^k, \text{istate}^{i-1})\}; \\ (\text{estate}^i, j^i, x_{ap}^i, x_{au}^i, \text{stop}^i) &\leftarrow E(1^k, \text{estate}^{i-1}, x_{hp}^i, x_{hu}^i, y_p^{i-1}, y_u^{i-1}); \\ (\text{astate}^i, y_s^i, y_p^i, y_u^i) &\leftarrow \text{Alg}(\text{astate}^{i-1}, x_{hs}^i, x_{hp}^i, x_{hu}^i, x_{ap}^i, x_{au}^i); \\ (\text{sstate}^i, \text{ustate}^i) &\leftarrow S_2^{U'(\text{ustate}^{i-1})}(\text{sstate}^{i-1}, x_{hu}^i, x_{au}^i) : (\text{ustate}^i); \\ \text{UVIEW}_{\text{ideal}} &= \text{UVIEW}_{\text{ideal}}^i \text{ if } \text{stop}^i = \text{TRUE}. \end{aligned}$$

在 ideal process 中, 模拟器 S_2 只能得到不受保护的输入 (x_{hu}^i, x_{au}^i) , 且允许访问 U' . U' 同 Pair one 一样保存自己的状态.

假设 T^U 是 Alg 的一次正确运行, 我们有如下定义.

定义 3 (α -efficient) 对任意输入 x , 如果 T 的运行时间不大于 Alg 的 α 倍, 则称 (T, U) 是 α -efficient.

定义 4 (β -checkable) 对任意输入 x , 如果 T 能以不小于 β 的概率检测到服务器的错误, 则称 (T, U) 是 β -checkable.

定义 5 ((α, β) -outsource-security) 如果 (T, U) 同时满足 α -efficient 和 β -checkable, 则称 (T, U) 是算法 Alg 的 (α, β) -outsource-secure 运行.

本文所提算法基于两个不可信、且不能相互勾结的服务器^[5]. 在这个模型中, 恶意环境 E 撰写两个软件 $U' = (U'_1, U'_2)$, T 安装这两个软件, 且 E 与 U'_1, U'_2 之间的通信必须通过 T . 因此, 攻击 T 的敌手定义为 $A = (E, U'_1, U'_2)$. 我们假定至多一个软件是不诚实的, 但是不知道是哪一个, 这种模型称为 one-malicious model. 在真实应用中, 这种模型相当于从两个供应商中买到两个不可信的软件, 只要他们之中至少一个是诚实的, 所提算法就是外包安全的^[1].

3 单个双线性对运算外包算法

同文献 [5], 所提算法需要执行子程序 Rand. Rand 的输入为大素数 q , 两个阶为 q 的加法循环群 G 和 $\hat{G}$, 双线性映射 $e: G \times \hat{G} \rightarrow G_T$, 其中 G_T 为阶为 q 的乘法循环群, 每次调用输出为随机向量:

$$(a^{-1}, b^{-1}, aP, b\hat{P}, a_2P, b_2\hat{P}, e(aP, b\hat{P}), e(a_2P, b\hat{P})^{-1}, e(aP, b_2\hat{P})^{-1}),$$

其中 $a, b, a_2, b_2 \in Z_q$.

3.1 具体算法

基于 one-malicious model, 本文提出了单个双线性对运算的安全外包算法 VBP. 在 VBP 算法中, T 调用子程序 Rand, 将双线性对运算外包给服务器 U_1 和 U_2 . 在 VBP 算法中, U_1 和 U_2 不能获得输入和输出的任何信息.

令 q 为一个素数. VBP 的输入为 $A \in G, B \in \hat{G}$, 输出为 $e(A, B)$. A, B 和 $e(A, B)$ 对 U_1 和 U_2 都是保密的. VBP 算法如下:

(1) 用户 T 首先执行两次 Rand 程序, 得到如下两个向量:

$$(a^{-1}, b^{-1}, aP, b\hat{P}, a_2P, b_2\hat{P}, e(aP, b\hat{P}), e(a_2P, b\hat{P})^{-1}, e(aP, b_2\hat{P})^{-1}), \\ (a_1^{-1}, b_1^{-1}, a_1P, b_1\hat{P}, a_3P, b_3\hat{P}, e(a_1P, b_1\hat{P}), e(a_3P, b_1\hat{P})^{-1}, e(a_1P, b_3\hat{P})^{-1}),$$

其中 $a, b, a_1, b_1, a_2, b_2, a_3, b_3 \in Z_q^*$.

(2) T 以随机顺序询问 U_1 :

$$U_1(A - aP, B - b\hat{P}) \rightarrow \alpha_{11} = e(A - aP, B - b\hat{P}), \\ U_1(A - aP + a_3P, b_1\hat{P}) \rightarrow \alpha_{12} = e(A - aP + a_3P, b_1\hat{P}), \\ U_1(a_1P, B - b\hat{P} + b_3\hat{P}) \rightarrow \alpha_{13} = e(a_1P, B - b\hat{P} + b_3\hat{P}).$$

同样, T 随机询问 U_2 如下:

$$U_2(A - aP + a_1P, B - b\hat{P} + b_1\hat{P}) \rightarrow \alpha_{21} = e(A - aP + a_1P, B - b\hat{P} + b_1\hat{P}), \\ U_2(A - aP + a_2P, b\hat{P}) \rightarrow \alpha_{22} = e(A - aP + a_2P, b\hat{P}), \\ U_2(aP, B - b\hat{P} + b_2\hat{P}) \rightarrow \alpha_{23} = e(aP, B - b\hat{P} + b_2\hat{P}).$$

(3) 从 U_1 和 U_2 分别收到 $\alpha_{11}, \alpha_{12}, \alpha_{13}$ 和 $\alpha_{21}, \alpha_{22}, \alpha_{23}$ 后, T 计算

$$\alpha_{12} \cdot e(a_3P, b_1\hat{P})^{-1} = e(A - aP, b_1\hat{P}), \alpha_{13} \cdot e(a_1P, b_3\hat{P})^{-1} = e(a_1P, B - b\hat{P}), \\ \alpha_{22} \cdot e(a_2P, b\hat{P})^{-1} = e(A - aP, b\hat{P}), \alpha_{23} \cdot e(aP, b_2\hat{P})^{-1} = e(aP, B - b\hat{P}).$$

然后, T 随机选择 $t_1, t_2 \in Z_q^*$, 随机询问 U_1 :

$$U_1(e(A - aP, b\hat{P}), t_1b^{-1}) \rightarrow \alpha_{14} = e(A - aP, t_1\hat{P}), \\ U_1(e(aP, B - b\hat{P}), t_2a^{-1}) \rightarrow \alpha_{15} = e(t_2P, B - b\hat{P}).$$

同样地, T 随机询问 U_2 :

$$U_2(e(A - aP, b_1\hat{P}), t_1b_1^{-1}) \rightarrow \alpha_{24} = e(A - aP, t_1\hat{P}), \\ U_2(e(a_1P, B - b\hat{P}), t_2a_1^{-1}) \rightarrow \alpha_{25} = e(t_2P, B - b\hat{P}).$$

(4) 最后, T 验证 U_1 和 U_2 的计算结果是否正确:

$$\alpha_{14} = \alpha_{24}, \alpha_{15} = \alpha_{25}, \alpha_{21} = \alpha_{11} \cdot e(A - aP, b_1\hat{P})e(a_1P, B - b\hat{P})e(a_1P, b_1\hat{P}).$$

如果不能通过, T 输出 “error”; 否则, T 计算

$$e(A, B) = \alpha_{11} \cdot e(A - aP, b\hat{P})e(aP, B - b\hat{P})e(aP, b\hat{P}).$$

Correctness. 假定两个服务器都是诚实的, 外包结果能被成功验证. 验证过程如下:

$$\alpha_{14} = U_1(\alpha_{22}e(a_2P, b\hat{P})^{-1}, t_1b^{-1}) = e(A - aP, t_1\hat{P}) = U_2(\alpha_{12}e(a_3P, b_1\hat{P})^{-1}, t_1b_1^{-1}) = \alpha_{24}, \\ \alpha_{15} = U_1(\alpha_{23}e(aP, b_2\hat{P})^{-1}, t_2a^{-1}) = e(t_2P, B - b\hat{P}) = U_2(\alpha_{13}e(a_1P, b_3\hat{P})^{-1}, t_2a_1^{-1}) = \alpha_{25},$$

$$\alpha_{21} = e(A - aP, b - B\hat{P})e(A - aP, b_1\hat{P})e(a_1P, B - b\hat{P})e(a_1P, b_1\hat{P}) \\ = \alpha_{11}e(A - aP, b_1\hat{P})e(a_1P, B - b\hat{P})e(a_1P, b_1\hat{P}).$$

注 1 在 one-malicious model 中, 因为 $a, b, a_1, b_1, a_2, b_2, a_3, b_3, t_1, t_2$ 是从 Z_q^* 随机选择的, 且对两个服务器是保密的, 下式

$$\begin{aligned}\alpha_{14} &= U_1(\alpha_{22}e(a_2P, b\hat{P})^{-1}, t_1b^{-1}) = U_2(\alpha_{12}e(a_3P, b_1\hat{P})^{-1}, t_1b_1^{-1}) = \alpha_{24}, \\ \alpha_{15} &= U_1(\alpha_{23}e(aP, b_2\hat{P})^{-1}, t_2a^{-1}) = U_2(\alpha_{13}e(a_1P, b_3\hat{P})^{-1}, t_2a_1^{-1}) = \alpha_{25}, \\ \alpha_{21} &= \alpha_{11} \cdot e(A - aP, b_1\hat{P})e(a_1P, B - b\hat{P})e(a_1P, b_1\hat{P})\end{aligned}$$

表明 U_1 和 U_2 生成了正确的 $\alpha_{11}, \alpha_{12}, \alpha_{13}$ 和 $\alpha_{21}, \alpha_{22}, \alpha_{23}$. 由于 $a, b, a_1, b_1, a_2, b_2, a_3, b_3, t_1, t_2$ 的随机性, VBP 算法能以概率 1 验证计算结果, 而文献 [21, 22] 的可验证性概率分别是 $1/2$ 和 $(1 - \frac{1}{3s})^2$, 其中 s 是一个小的整数. 对用户来说, 需要 G_1 或 G_2 群中的 8 次点加, G_T 群中的 14 次模乘, 并调用两次 Rand 子程序. 在文献 [17] 的算法中, 尽管可验证概率为 1, 但用户需要执行一些昂贵的运算, 如点乘、模指数运算等. 因此, 同文献 [17, 21, 22] 中的算法相比, VBP 算法能同时提高单个双线性对运算的可验证概率和实现效率.

注 2 VBP 算法同样适用于 G 和 $\hat{G}$ 是乘法群的情况. 令 $g, \hat{g}$ 分别是 $G, \hat{G}$ 的生成元, $e: G \times \hat{G} \rightarrow G_T$ 是一个双线性映射. VBP 的输入是 $A \in G, B \in \hat{G}$, 输出是 $e(A, B)$. 具体过程如下:

(1) T 首先调用两次 Rand 程序, 生成两个向量如下:

$$\begin{aligned}(a^{-1}, b^{-1}, g^a, \hat{g}^b, g^{a_2}, \hat{g}^{b_2}, e(g^a, \hat{g}^b), e(g^{a_2}, \hat{g}^b)^{-1}, e(g^a, \hat{g}^{b_2})^{-1}), \\ (a_1^{-1}, b_1^{-1}, g^{a_1}, \hat{g}^{b_1}, g^{a_3}, \hat{g}^{b_3}, e(g^{a_1}, \hat{g}^{b_1}), e(g^{a_3}, \hat{g}^{b_1})^{-1}, e(g^{a_1}, \hat{g}^{b_3})^{-1}),\end{aligned}$$

其中 $a, b, a_1, b_1, a_2, b_2, a_3, b_3 \in Z_q^*$.

(2) T 以随机顺序询问 U_1 如下:

$$\begin{aligned}U_1(A \cdot g^{-a}, B \cdot \hat{g}^{-b}) &\rightarrow \alpha_{11} = e(A \cdot g^{-a}, B \cdot \hat{g}^{-b}), \\ U_1(A \cdot g^{-a}g^{a_3}, \hat{g}^{b_1}) &\rightarrow \alpha_{12} = e(A \cdot g^{-a}g^{a_3}, \hat{g}^{b_1}), \\ U_1(g^{a_1}, B \cdot \hat{g}^{-b}\hat{g}^{b_3}) &\rightarrow \alpha_{13} = e(g^{a_1}, B \cdot \hat{g}^{-b}\hat{g}^{b_3}).\end{aligned}$$

同样地, T 以随机顺序询问 U_2 :

$$\begin{aligned}U_2(A \cdot g^{-a}g^{a_1}, B \cdot \hat{g}^{-b}\hat{g}^{b_1}) &\rightarrow \alpha_{21} = e(A \cdot g^{-a}g^{a_1}, B \cdot \hat{g}^{-b}\hat{g}^{b_1}), \\ U_2(A \cdot g^{-a}g^{a_2}, \hat{g}^b) &\rightarrow \alpha_{22} = e(A \cdot g^{-a}g^{a_2}, \hat{g}^b), \\ U_2(g^a, B \cdot \hat{g}^{-b}\hat{g}^{b_2}) &\rightarrow \alpha_{23} = e(g^a, B \cdot \hat{g}^{-b}\hat{g}^{b_2}).\end{aligned}$$

(3) 从 U_1, U_2 处分别收到 $\alpha_{11}, \alpha_{12}, \alpha_{13}$ 和 $\alpha_{21}, \alpha_{22}, \alpha_{23}$ 后, T 计算

$$\begin{aligned}\alpha_{12} \cdot e(g^{a_3}, \hat{g}^{b_1})^{-1} &= e(A \cdot g^a, \hat{g}^{b_1}), \alpha_{13} \cdot e(g^{a_1}, \hat{g}^{b_3})^{-1} = e(g^{a_1}, B \cdot \hat{g}^{-b}), \\ \alpha_{22} \cdot e(g^{a_2}, \hat{g}^b)^{-1} &= e(A \cdot g^a, \hat{g}^b), \alpha_{23} \cdot e(g^a, \hat{g}^{b_2})^{-1} = e(g^a, B \cdot \hat{g}^{-b}).\end{aligned}$$

然后, T 随机选择 $t_1, t_2 \in Z_q^*$, 以随机顺序询问 U_1 如下:

$$\begin{aligned}U_1(e(A \cdot g^{-a}, \hat{g}^b), t_1b^{-1}) &\rightarrow \alpha_{14} = e(A \cdot g^{-a}, \hat{g}^{t_1}), \\ U_1(e(g^a, B \cdot \hat{g}^{-b}), t_2a^{-1}) &\rightarrow \alpha_{15} = e(g^{t_2}, B \cdot \hat{g}^{-b}).\end{aligned}$$

同样地, T 以随机顺序询问 U_2 :

$$\begin{aligned}U_2(e(A \cdot g^{-a}, \hat{g}^{b_1}), t_1b_1^{-1}) &\rightarrow \alpha_{24} = e(A \cdot g^{-a}, \hat{g}^{t_1}), \\ U_2(e(g^{a_1}, B \cdot \hat{g}^{-b}), t_2a_1^{-1}) &\rightarrow \alpha_{25} = e(g^{t_2}, B \cdot \hat{g}^{-b}).\end{aligned}$$

(4) 最终, T 验证 U_1 和 U_2 的计算结果是否正确:

$$\alpha_{14} = \alpha_{24}, \alpha_{15} = \alpha_{25}, \alpha_{21} = \alpha_{11} \cdot e(A \cdot g^{-a}, \hat{g}^{b_1})e(g^{a_1}, B \cdot \hat{g}^{-b})e(g^{a_1}, \hat{g}^{b_1}).$$

如果不能通过, T 输出 “error”; 否则, T 计算

$$e(A, B) = \alpha_{11} \cdot e(A \cdot g^{-a}, \hat{g}^b)e(g^a, B \cdot \hat{g}^{-b})e(g^a, \hat{g}^b).$$

3.2 安全性证明

定理 3.1 在 one-malicious model 中, 所提算法 $(T, (U_1, U_2))$ 是 VBP 算法的正确运行, 其中 (A, B) 可能是诚实的, 保密的; 诚实的, 受保护的; 或恶意的, 受保护的.

证明 在 one-malicious model 中, 令 $A = (E, U'_1, U'_2)$ 是与 PPT 算法 T 交互的 PPT 敌手.

首先证明 $\text{EVIEW}_{\text{real}} \sim \text{EVIEW}_{\text{ideal}}$, 即在算法 $(T, (U_1, U_2))$ 执行过程中, 恶意环境 E 不能获得输入和输出的任何信息. 如果输入 (A, B) 是诚实的, 受保护的; 或者恶意的, 受保护的, 模拟器 S_1 同在 real 环境中一样运行. 因此, 只需证明 (A, B) 是诚实的, 保密的情况.

假定 (A, B) 是诚实的, 保密的输入. 模拟器 S_1 在 ideal 环境中运行如下. 当收到第 i 轮的输入时, S_1 忽略它们, 并随机选择 3 个向量 (α'_j, β'_j) 发送给 U'_1 和 U'_2 . 收到 U'_1 和 U'_2 的输出后, S_1 随机选择两个向量 (α_j, β_j) , 并发送给 U'_1 和 U'_2 . 最终, S_1 检查 3 个输出 $e(\alpha'_j, \beta'_j)$ 和两个输出 $\alpha_j^{\beta_j}$. 如果检查到一个错误, S_1 保存所有的状态, 并输出 $Y_p^i = \text{“error”}$, $Y_u^i = \emptyset$, $\text{rep}^i = 1$, ideal 进程的最终输出是 $(\text{estate}^i, \text{“error”}, \emptyset)$; 否则, S_1 输出 $Y_p^i = \emptyset$, $Y_u^i = \emptyset$, $\text{rep}^i = 0$, ideal 进程的最终输出是 $(\text{estate}^i, y_p^i, y_u^i)$.

下面说明, 在 real 环境下发给 (U'_1, U'_2) 的输入和 ideal 环境下的输入是计算上不可区分的. 在 ideal 环境中, 输入是随机选择且均匀分布的. 而在 real 环境中, T 的 5 个询问都是通过调用 Rand 子程序生成的, 因此是随机的. 如果 (U'_1, U'_2) 在第 i 轮是诚实的, $\text{EVIEW}_{\text{real}}^i \sim \text{EVIEW}_{\text{ideal}}^i$, VBP 的输出是真实的, $\text{rep}^i = 0$. 如果 (U'_1, U'_2) 在第 i 轮有一个是不诚实的, 错误一定能被 T 和 S_1 以概率 1 检查出来, 并输出 “error”. 因此, 在第 i 轮, $\text{EVIEW}_{\text{real}}^i \sim \text{EVIEW}_{\text{ideal}}^i$, 最终 $\text{EVIEW}_{\text{real}} \sim \text{EVIEW}_{\text{ideal}}$.

第二, 证明 $\text{UVIEW}_{\text{real}} \sim \text{UVIEW}_{\text{ideal}}$, 即在算法 $(T, (U_1, U_2))$ 执行过程中, 恶意软件 (U'_1, U'_2) 不能获得输入和输出的任何信息. 在 ideal 环境中, 模拟器 S_2 运行如下: 当收到第 i 轮的输入时, S_2 忽略它们, 并随机选择 3 个向量 (α'_j, β'_j) 发送给 U'_1 和 U'_2 . 收到 U'_1 和 U'_2 的输出后, S_2 随机选择两个向量 (α_j, β_j) , 并发送给 U'_1 和 U'_2 . 最终, S_2 保存它和 (U'_1, U'_2) 的状态. 由于诚实的, 保密的; 诚实的, 受保护的; 恶意的, 受保护的输入对 U'_1 和 U'_2 都是保密的, 因此 S_2 适用于这 3 种情况. 如上所述, real 环境下发给 (U'_1, U'_2) 的输入和 ideal 环境下 S_2 的输入是计算上不可区分的. 因此第 i 轮, $\text{UVIEW}_{\text{real}}^i \sim \text{UVIEW}_{\text{ideal}}^i$, 最终 $\text{UVIEW}_{\text{real}} \sim \text{UVIEW}_{\text{ideal}}$.

定理 3.2 所提算法 $(T, (U_1, U_2))$ 是完全验证的, 即用户能以概率 1 检测到服务器的错误结果.

证明 假定 U_1 是诚实的服务器, U_2 是恶意的服务器. 在算法最后, 用户验证外包结果如下:

$$\alpha_{14} = U_1(\alpha_{22}e(a_2P, b_1\hat{P})^{-1}, t_1b^{-1}) = U_2(\alpha_{12}e(a_3P, b_1\hat{P})^{-1}, t_1b_1^{-1}) = \alpha_{24}, \quad (1)$$

$$\alpha_{15} = U_1(\alpha_{23}e(aP, b_2\hat{P})^{-1}, t_2a^{-1}) = U_2(\alpha_{13}e(a_1P, b_3\hat{P})^{-1}, t_2a_1^{-1}) = \alpha_{25}, \quad (2)$$

$$\alpha_{21} = \alpha_{11}\alpha_{12} \cdot e(a_3P, b_1\hat{P})^{-1}\alpha_{13}e(a_1P, b_3\hat{P})^{-1}e(a_1P, b_1\hat{P}). \quad (3)$$

由于 U_1 是诚实的服务器, 一定会返回真实的 $\alpha_{11}, \alpha_{12}, \alpha_{13}$, 用户计算

$$\alpha_{12} \cdot e(a_3P, b_1\hat{P})^{-1} = e(A - aP, b_1\hat{P}), \alpha_{13} \cdot e(a_1P, b_3\hat{P})^{-1} = e(a_1P, B - b\hat{P}),$$

U_2 返回: $\alpha_{21} = e(A - aP + a_1P, B - b\hat{P} + b_1\hat{P}) = \alpha_{11}e(A - aP, b_1\hat{P})e(a_1P, B - b\hat{P})e(a_1P, b_1\hat{P})$.

否则, 式 (3) 不能通过验证. 因此, U_2 只能在式 (1) 和 (2) 中欺骗用户.

假定 U_2 返回 α_{22}, α_{23} 的错误值, 我们记为 $\overline{\alpha_{22}}, \overline{\alpha_{23}}$. 收到 U_1 和 U_2 返回的计算结果后, T 计算 $\overline{\alpha_{22}} \cdot e(a_2P, b\hat{P})^{-1}, \overline{\alpha_{23}} \cdot e(aP, b_2\hat{P})^{-1}$, 然后随机选择 $t_1, t_2 \in Z_q^*$ 并询问

$$U_1(\overline{\alpha_{22}} \cdot e(a_2P, b\hat{P})^{-1}, t_1b^{-1}), U_1(\overline{\alpha_{23}} \cdot e(aP, b_2\hat{P})^{-1}, t_2a^{-1}), \\ U_2(\alpha_{12} \cdot e(a_3P, b_1\hat{P})^{-1}, t_1b_1^{-1}), U_2(\alpha_{13} \cdot e(a_1P, b_3\hat{P})^{-1}, t_2a_1^{-1}).$$

为了通过式 (1) 和 (2) 的验证, U_2 必须返回和 U_1 一样的计算结果. 因 U_1 是诚实的服务器, 计算

$$\alpha_{14} = U_1(\overline{\alpha_{22}} \cdot e(a_2P, b\hat{P})^{-1}, t_1b^{-1}) = [\overline{\alpha_{22}} \cdot e(a_2P, b\hat{P})^{-1}]^{t_1b^{-1}}, \\ \alpha_{15} = U_1(\overline{\alpha_{23}} \cdot e(aP, b_2\hat{P})^{-1}, t_2a^{-1}) = [\overline{\alpha_{23}} \cdot e(aP, b_2\hat{P})^{-1}]^{t_2a^{-1}}.$$

如果式 (1) 和 (2) 能通过验证, 即

$$\alpha_{14} = U_1(\overline{\alpha_{22}} \cdot e(a_2P, b\hat{P})^{-1}, t_1b^{-1}) = [\overline{\alpha_{22}} \cdot e(a_2P, b\hat{P})^{-1}]^{t_1b^{-1}} = U_2(\alpha_{12} \cdot e(a_3P, b_1\hat{P})^{-1}, t_1b_1^{-1}) = \overline{\alpha_{24}}, \\ \alpha_{15} = U_1(\overline{\alpha_{23}} \cdot e(aP, b_2\hat{P})^{-1}, t_2a^{-1}) = [\overline{\alpha_{23}} \cdot e(aP, b_2\hat{P})^{-1}]^{t_2a^{-1}} = U_2(\alpha_{13} \cdot e(a_1P, b_3\hat{P})^{-1}, t_2a_1^{-1}) = \overline{\alpha_{25}}.$$

U_2 必须返回

$$\overline{\alpha_{24}} = U_2(\alpha_{12} \cdot e(a_3P, b_1\hat{P})^{-1}, t_1b_1^{-1}) = [\overline{\alpha_{22}} \cdot e(a_2P, b\hat{P})^{-1}]^{t_1b^{-1}}, \\ \overline{\alpha_{25}} = U_2(\alpha_{13} \cdot e(a_1P, b_3\hat{P})^{-1}, t_2a_1^{-1}) = [\overline{\alpha_{23}} \cdot e(aP, b_2\hat{P})^{-1}]^{t_2a^{-1}}.$$

因此, U_2 必须知道 b_1b^{-1}, a_1a^{-1} , 计算 $t_1b^{-1} = (t_1b_1^{-1}) \cdot (b_1b)^{-1}, t_2a^{-1} = (t_2a_1^{-1}) \cdot (a_1a)^{-1}$, 然后伪造 $\overline{\alpha_{22}}, \overline{\alpha_{23}}, \overline{\alpha_{24}}, \overline{\alpha_{25}}$. 如 3.1 小节所示, a, b, a_1, b_1 是通过 Rand 子程序从 Z_q^* 中随机选择的, b_1b^{-1}, a_1a^{-1} 对 U_2 是保密且随机的. 因此, 恶意的服务器不能返回错误的计算结果并同时通过式 (1), (2) 和 (3) 的验证, VBP 算法能以概率 1 验证外包结果的正确性.

定理 3.3 在 one-malicious model 中, 所提算法 $(T, (U_1, U_2))$ 是 VBP 算法的 $(O(\frac{1}{m}), 1)$ 安全运行, 其中 m 是 $G, \hat{G}$ 的阶 q 的比特长度.

证明 为了计算 $e(A, B)$, VBP 算法需要调用两次 Rand 子程序, 执行 8 次 G 或 $\hat{G}$ 中的点加运算, 14 次 $\hat{G}$ 中的模乘运算. 如文献 [21] 所示, 一次双线性对运算需要 $O(m)$ 次有限域中的模乘运算, 其中 m 为 q 的比特长度. 因此, 所提算法是双线性对运算的 $O(\frac{1}{m})$ -运行. 另一方面, 由定理 3.2, 如果 U_1 或 U_2 返回不正确的计算结果, 用户能以概率 1 检测出错误.

3.3 算法比较

表 1 比较了隐私保护的单个双线性对运算外包算法, 其中 s 是小的整数. “PA, SM, M, MInv, MExp” 分别代表 G 或 $\hat{G}$ 中的点加、标量乘、 G_T 中的模乘、模逆和模指数运算.

从表 1 中可以看出, 尽管 BJN 算法^[17] 基于单个不可信的服务器, 但是它对单个双线性对运算没有任何优势, 需要用户执行 10 次 G_T 中的模指数运算, 6 次 G 或 $\hat{G}$ 中的标量乘运算. 同文献 [21, 22] 中的算法相比, VBP 算法提高了外包算法的可验证概率, 能有效地验证外包结果的正确性. 尽管增加了一些计算代价, 但 VBP 算法的可验证概率为 1. 而在文献 [21, 22] 中, 可验证概率分别是 $1/2$ 和 $(1 - \frac{1}{3s})^2$.

表 1 单个双线性对运算外包算法比较

Table 1 Comparison among outsourcing algorithms of single bilinear pairing

Algorithm	BJN ^[17]	Pair ^[21]	TZR1 ^[22]	TZR2 ^[22]	VBP
PA(T)	4	5	4	$O(\log s)$	8
M(T)	6	4	3	$O(\log s)$	14
MExp(T)	10	0	0	0	0
SM(T)	6	0	0	0	0
Rand	0	9SM+3Pair	8SM+2MExp +2MInv+5M	8SM+2Exp +4MInv+8M	8SM+6Pair +8MInv
Pair(U)	4	8	6	6	6
MExp(U)	0	0	0	0	4
Servers	One	Two	Two	Two	Two
Checkability	1	1/2	1/2	$(1 - \frac{1}{3s})^2$	1
Communication rounds	4	2	2	2	4

4 多个双线性对运算外包算法

本节提出多个双线性对运算 $\prod_{i=1}^n e(A_i, B_i) (n \geq 2)$ 的外包算法, 其在密码方案中有广泛应用, 如基于属性加密^[27]、双系统加密^[28]、基于身份的匿名加密^[29]等.

4.1 具体算法

基于 one-malicious model, 本文提出多个双线性对运算外包算法 VMBP. 令 q 为一个大素数, VMBP 的输入是 $A_1, \dots, A_n \in G, B_1, \dots, B_n \in \hat{G}$, 输出是 $\prod_{i=1}^n e(A_i, B_i)$, 其中所有的 $A_i, B_i (1 \leq i \leq n)$ 对 U_1 和 U_2 都是保密的.

同第 3 节一样, VMBP 算法需要调用子程序 Rand, 其输入为素数 q , 两个加法循环群 $G, \hat{G}$ 和双线性映射 $e: G \times \hat{G} \rightarrow G_T$, 其中 G_T 为阶为 q 的乘法群, 输出为如下的随机向量:

$$(a^{-1}, b^{-1}, aP, b\hat{P}, a_2P, b_2\hat{P}, e(aP, b\hat{P})^n, e(a_2P, b\hat{P})^{-1}, e(aP, b_2\hat{P})^{-1}),$$

其中 $a, b, a_2, b_2 \in Z_q$.

所提的 VMBP 具体算法如下:

(1) 用户 T 首先执行两次 Rand 程序, 得到两个向量

$$(a^{-1}, b^{-1}, aP, b\hat{P}, a_2P, b_2\hat{P}, e(aP, b\hat{P})^n, e(a_2P, b\hat{P})^{-1}, e(aP, b_2\hat{P})^{-1}),$$

$$(a_1^{-1}, b_1^{-1}, a_1P, b_1\hat{P}, a_3P, b_3\hat{P}, e(a_1P, b_1\hat{P})^n, e(a_3P, b_1\hat{P})^{-1}, e(a_1P, b_3\hat{P})^{-1}),$$

其中 $a, b, a_1, b_1, a_2, b_2, a_3, b_3 \in Z_q^*$.

(2) T 以随机顺序询问 U_1 :

$$U_1(A_1 - aP, B_1 - b\hat{P}) \rightarrow \alpha_{111} = e(A_1 - aP, B_1 - b\hat{P}),$$

$$\dots,$$

$$U_1(A_n - aP, B_n - b\hat{P}) \rightarrow \alpha_{11n} = e(A_n - aP, B_n - b\hat{P}),$$

$$U_1\left(\sum_{i=1}^n(A_i - aP) + a_3P, b_1\hat{P}\right) \rightarrow \alpha_{12} = e\left(\sum_{i=1}^n(A_i - aP) + a_3P, b_1\hat{P}\right),$$

$$U_1\left(a_1P, \sum_{i=1}^n(B_i - b\hat{P}) + b_3\hat{P}\right) \rightarrow \alpha_{13} = e\left(a_1P, \sum_{i=1}^n(B_i - b\hat{P}) + b_3\hat{P}\right).$$

同样, T 随机询问 U_2 如下:

$$U_2(A_1 - aP + a_1P, B_1 - b\hat{P} + b_1\hat{P}) \rightarrow \alpha_{211} = e(A_1 - aP + a_1P, B_1 - b\hat{P} + b_1\hat{P}),$$

$$\dots,$$

$$U_2(A_n - aP + a_1P, B_n - b\hat{P} + b_1\hat{P}) \rightarrow \alpha_{21n} = e(A_n - aP + a_1P, B_n - b\hat{P} + b_1\hat{P}),$$

$$U_2\left(\sum_{i=1}^n(A_i - aP) + a_2P, b\hat{P}\right) \rightarrow \alpha_{22} = e\left(\sum_{i=1}^n(A_i - aP) + a_2P, b\hat{P}\right),$$

$$U_2\left(aP, \sum_{i=1}^n(B_i - b\hat{P}) + b_2\hat{P}\right) \rightarrow \alpha_{23} = e\left(aP, \sum_{i=1}^n(B_i - b\hat{P}) + b_2\hat{P}\right).$$

(3) 从 U_1 和 U_2 处分别收到 $\alpha_{111}, \dots, \alpha_{11n}, \alpha_{12}, \alpha_{13}$ 和 $\alpha_{211}, \dots, \alpha_{21n}, \alpha_{22}, \alpha_{23}$ 后, T 计算

$$\alpha_{11} = \prod_{i=1}^n \alpha_{11i}, \alpha_{21} = \prod_{i=1}^n \alpha_{21i},$$

$$\alpha_{12} \cdot e(a_3P, b_1\hat{P})^{-1} = e\left(\sum_{i=1}^n(A_i - aP), b_1\hat{P}\right), \quad \alpha_{13} \cdot e(a_1P, b_3\hat{P})^{-1} = e\left(a_1P, \sum_{i=1}^n(B_i - b\hat{P})\right),$$

$$\alpha_{22} \cdot e(a_2P, b\hat{P})^{-1} = e\left(\sum_{i=1}^n(A_i - aP), b\hat{P}\right), \quad \alpha_{23} \cdot e(aP, b_2\hat{P})^{-1} = e\left(aP, \sum_{i=1}^n(B_i - b\hat{P})\right).$$

然后, T 随机选择 $t_1, t_2 \in Z_q^*$, 随机询问 U_1 :

$$U_1\left(e\left(\sum_{i=1}^n(A_i - aP), b\hat{P}\right), t_1b^{-1}\right) \rightarrow \alpha_{14} = e\left(\sum_{i=1}^n(A_i - aP), t_1\hat{P}\right),$$

$$U_1\left(e\left(aP, \sum_{i=1}^n(B_i - b\hat{P})\right), t_2a^{-1}\right) \rightarrow \alpha_{15} = e\left(t_2P, \sum_{i=1}^n(B_i - b\hat{P})\right).$$

同样地, T 随机询问 U_2 :

$$U_2\left(e\left(\sum_{i=1}^n(A_i - aP), b_1\hat{P}\right), t_1b_1^{-1}\right) \rightarrow \alpha_{24} = e\left(\sum_{i=1}^n(A_i - aP), t_1\hat{P}\right),$$

$$U_2\left(e\left(a_1P, \sum_{i=1}^n(B_i - b\hat{P})\right), t_2a_1^{-1}\right) \rightarrow \alpha_{25} = e\left(t_2P, \sum_{i=1}^n(B_i - b\hat{P})\right).$$

(4) 最后, T 验证 U_1 和 U_2 的计算结果是否正确:

$$\alpha_{14} = \alpha_{24}, \alpha_{15} = \alpha_{25}, \alpha_{21} = \alpha_{11} \cdot e\left(\sum_{i=1}^n(A_i - aP), b_1\hat{P}\right) e\left(a_1P, \sum_{i=1}^n(B_i - b\hat{P})\right) e(a_1P, b_1\hat{P})^n.$$

如果不能通过, T 输出 “error”; 否则, T 计算

$$\prod_{i=1}^n e(A_i, B_i) = \alpha_{11} \cdot e\left(\sum_{i=1}^n(A_i - aP), b\hat{P}\right) e\left(aP, \sum_{i=1}^n(B_i - b\hat{P})\right) e(aP, b\hat{P})^n.$$

注 3 实际上, n 个双线性对 $\prod_{i=1}^n e(A_i, B_i)$ 的外包可通过调用 n 次单个双线性对 $e(A, B)$ 进行, 需要用户执行 G 或 $\hat{G}$ 群中的 $8n$ 次点加, G_T 群中的 $14n$ 次模乘, 而 VMBP 算法只需执行 $6n + 2$ 次点加和 $2n + 12$ 次模乘. 对服务器来说, 调用 n 次 VBP 算法需要执行 $6n$ 次双线性对运算, $4n$ 次 G_T 群中的模指数运算, 而在 VMBP 算法中, 用户只需执行 $2n + 2$ 次双线性对运算和 4 次模指数运算. 因此, 当 $n \geq 2$ 时, VMBP 算法比调用 n 次 VBP 算法效率更高.

注 4 同注 2 一样, VMBP 算法也适用于 $G, \hat{G}$ 为两个乘法群的情况.

4.2 安全性证明

定理 4.1 在 one-malicious model 中, 4.1 小节中所提算法 $(T, (U_1, U_2))$ 是 VMBP 算法的正确运行, 其中 $(A_1, \dots, A_n; B_1, \dots, B_n)$ 可能是诚实的, 保密的; 诚实的, 受保护的; 或恶意的, 受保护的.

定理 4.2 在 one-malicious model 中, 4.1 小节中所提算法 $(T, (U_1, U_2))$ 是 VMBP 算法的 $(O(\frac{1}{m}), 1)$ 安全运行, 即用户能以概率 1 检测到服务器的错误.

5 外包算法在密码学中的应用

本节介绍 VBP 和 VMBP 算法的应用, 包括基于身份的匿名加密 (AIBE) 方案^[29], 及基于身份签名 (IBS) 方案^[30]. 令 $G, \hat{G}, G_T$ 是 3 个乘法循环群, $g, \hat{g}$ 分别是 $G, \hat{G}$ 的生成元, $e: G \times \hat{G} \rightarrow G_T$ 是一个双线性映射. 在下面的方案中, $G = \hat{G}$.

5.1 包含外包解密算法的 Boyen-Waters AIBE 方案

Setup. 随机选择生成元 $g \in G$, 随机群元素 $g_0, g_1 \in G$, 以及随机指数 $\omega, t_1, t_2, t_3, t_4 \in Z_p$, 则 PKG (private key generator) 私钥 $\text{MSK} = \{\omega, t_1, t_2, t_3, t_4\}$, 公钥定义为

$$\text{PK} = \{e(g, g)^{t_1 t_2 \omega}, g, g_0, g_1, v_1 = g^{t_1}, v_2 = g^{t_2}, v_3 = g^{t_3}, v_4 = g^{t_4}\}.$$

Extract(MSK, ID). 对于身份 ID, PKG 随机选择 $r_1, r_2 \in Z_p$, 计算私钥 SK_{ID} 如下:

$$\begin{aligned} d_0 &= g^{r_1 t_1 t_2 + r_2 t_3 t_4}, & d_1 &= g^{-\omega t_2} (g_0 g_1^{\text{ID}})^{-r_1 t_2}, & d_2 &= g^{-\omega t_2} (g_0 g_1^{\text{ID}})^{-r_1 t_1}, \\ d_3 &= (g_0 g_1^{\text{ID}})^{-r_2 t_4}, & d_4 &= (g_0 g_1^{\text{ID}})^{-r_2 t_3}, & \text{SK}_{\text{ID}} &= (d_0, d_1, d_2, d_3, d_4). \end{aligned}$$

Encrypt(PK, ID, M). 对于身份 ID, 消息 $M \in G_T$, 随机选择 $s, s_1, s_2 \in Z_p$, 生成如下密文:

$$\text{CT} = (C', C_0, C_1, C_2, C_3, C_4) = (Me(g, g)^{t_1 t_2 \omega}, (g_0 g_1^{\text{ID}})^s, v_1^{s-s_1}, v_2^{s_1}, v_3^{s-s_2}, v_4^{s_2}).$$

Decrypt(PK, ID, CT). 用户首先执行 VMBP 算法, 得到

$$\prod_{i=0}^4 e(C_i, d_i) = \text{VMBP}(C_0, C_1, C_2, C_3, C_4; d_0, d_1, d_2, d_3, d_4),$$

并计算 $C' \prod_{i=0}^4 e(C_i, d_i) = M$.

图 1 VBP 算法模拟实现

Figure 1 Simulation for VBP algorithm

图 2 VMBP 算法模拟实现 ($n=5$)Figure 2 Simulation for VMBP algorithm ($n=5$)

5.2 包含签名外包验证算法的 Paterson-Schuldt IBS 方案

Setup. 随机选择 $\alpha \in Z_p, g_2 \in G$, 计算 $g_1 = g^\alpha$. 然后, 选择 $u', m' \in G$, 及长度分别为 n_u, n_m 的向量 $U = (u_i), M = (m_i)$, 其中 u_i, m_i 是 G 中的元素. 公钥 $PK = (g, g_1, g_2, u', U, m', M)$, PKG 私钥为 g_2^α .

Extract. 令 u 为代表身份的长为 n_u 的比特串, $u[i]$ 为 u 的第 i bit. 令 $U \subset \{1, 2, \dots, n_u\}$ 为下标 i 的集合, 其中 $u[i] = 1$. 对于身份 u , PKG 随机选择 $r_u \in Z_p$, 计算私钥 $d_u = (g_2^\alpha (u' \prod_{i \in U} u_i)^{r_u}, g^{r_u})$.

Sign. 令 $M \subset \{1, 2, \dots, n_m\}$ 为下标 j 的集合, 其中 $m[j] = 1$, m 为一个消息, $m[j]$ 为 m 的第 j bit. 对于消息 m , 随机选择 $r_m \in Z_p$, 计算签名 $\sigma = (g_2^\alpha (u' \prod_{i \in U} u_i)^{r_u} (m' \prod_{j \in M} m_j)^{r_m}, g^{r_u}, g^{r_m})$.

Verify. 对于消息 m 关于身份 u 的签名 $\sigma = (V, R_u, R_m)$, 用户首先执行 VBP 算法, 得到 $(V, g) = \text{VBP}(V, g)$, 然后运行 VMBP 算法, 获得

$$e(g_2, g_1) e\left(u' \prod_{i \in U} u_i, R_u\right) e\left(m' \prod_{j \in M} m_j, R_m\right) = \text{VMBP}\left(g_2, u' \prod_{i \in U} u_i, m' \prod_{j \in M} m_j; g_1, R_u, R_m\right),$$

并验证 $e(V, g) = e(g_2, g_1) e(u' \prod_{i \in U} u_i, R_u) e(m' \prod_{j \in M} m_j, R_m)$ 是否成立.

6 性能分析

本节通过实验对 VBP 和 VMBP 算法进行性能分析. 用户和服务器分别使用 Intel Celeron Processor (1.2 GHz, 2 G 内存) 和 Intel Xeon Processor (3.4 GHz, 32 G 内存) 模拟. 编程语言为 Java, 使用 Java Pairing-Based Cryptography (JPBC) 数据库. 参数 q 为随机生成的 160-bit 的素数.

图 1 和 2 分别提供了 VBP 和 VMBP 算法 ($n=5$) 的模拟实现, 用户能以概率 1 检测到服务器的错误结果. 在计算单个和多个双线性对时, 由于部分计算转移给两个服务器, 用户 T 在外包计算中花费的时间远远小于直接计算的时间. 因此, VBP 和 VMBP 算法是可验证的安全的单个和多个双线性对运算外包算法.

表 2 比较了 VBP 算法和文献 [17, 21, 22] 提出的单个双线性对外包算法的计算时间. 从表 2 可以看出, 对于用户 T 来说, VBP 算法效率高于 BJN 和 TZR2 算法. 同 Pair 和 TZR1 算法相比, 增加了

表 2 单个双线性对运算外包算法性能比较

Table 2 Efficiency comparison among outsourcing algorithms of single bilinear pairing

	BJN ^[17]	Pair ^[21]	TZR1 ^[22]	TZR2 ^[22] ($s = 4$)	VBP
T (ms)	246	1.062	0.724	2.311	1.820
U_1 (ms)	23.067	23.074	10.506	17.469	20.033
U_2 (ms)	0	22.945	20.768	17.182	20.029

表 3 多个双线性对运算外包算法性能比较

Table 3 Efficiency comparison among outsourcing algorithms of multiple bilinear pairings

n	调用 n 次 VBP 算法			VMBP 算法		
	T (ms)	U_1 (ms)	U_2 (ms)	T (ms)	U_1 (ms)	U_2 (ms)
2	4.13	42.6	43.28	3.01	25.32	25.27
3	6.28	63.9	64.92	4.18	29.48	29.40
4	8.43	85.2	86.56	4.99	34.47	33.34
5	10.58	106.5	108.2	6.26	41.92	41.89
6	12.73	127.8	129.84	7.43	48.69	48.58
7	14.88	149.1	151.48	8.70	52.70	53.00
8	17.03	170.4	173.12	10.12	58.80	59.00
9	19.18	191.7	194.76	11.16	66.00	65.92
10	21.33	213	216.4	12.28	75.08	74.90

少量运算以提高可验证概率. 表 3 比较了 VMBP 算法和调用 n 次 VBP 算法时, 用户和服务器的计算时间. 因此, 对用户来说, VBP 和 VMBP 算法基于 one-malicious model, 同时提高了可验证概率和实现效率.

7 结论

本文分别针对单个和多个双线性对运算, 提出了两个可验证的外包计算算法. 基于两个不可信但不能互相勾结的服务器, 所提算法能够保护用户输入和输出的隐私性, 且用户能以概率 1 检测到服务器的错误结果. 同已有算法比较, 所提算法提高了外包算法的可验证概率, 且没有降低计算效率.

参考文献

- 1 Chen X F, Li J, Ma J F, et al. New algorithms for secure outsourcing of modular exponentiations. *IEEE Trans Parall Distrib Syst*, 2014, 25: 2386–2396
- 2 Gennaro R, Gentry C, Parno B. Non-interactive verifiable computing: outsourcing computation to untrusted workers. In: *Proceedings of the 30th Annual Conference on Advances in Cryptology*. Berlin: Springer, 2010. 465–482
- 3 Chung K, Kalai Y, Vadhan S. Improved delegation of computation using fully homomorphic encryption. In: *Proceedings of the 30th Annual Conference on Advances in Cryptology*. Berlin: Springer, 2010. 483–501
- 4 Chaum D, Pedersen T. Wallet databases with observers. In: *Proceedings of 12th Annual Conference on Advances in Cryptology*. Berlin: Springer, 1992. 89–105
- 5 Hohenberger S, Lysyanskaya A. How to securely outsource cryptographic computations. In: *Proceedings of the 2nd International Conference on Theory of Cryptography*. Berlin: Springer, 2005. 264–282

- 6 Green M, Hohenberger S, Waters B. Outsourcing the decryption of ABE ciphertexts. In: Proceedings of the 20th USENIX Conference on Security. New York: ACM, 2011. 34
- 7 Lai J Z, Deng R H, Guan C W, et al. Attribute-based encryption with verifiable outsourced decryption. *IEEE Trans Inf Foren Secur*, 2013, 8: 1343–1354
- 8 Golle P, Mironov I. Uncheatable distributed computations. In: Proceedings of the Cryptographers' Track at RSA Conference. Berlin: Springer, 2001. 425–440
- 9 Atallah M, Pantazopoulos K, Rice J, et al. Secure outsourcing of scientific computations. *Adv Comput*, 2001, 54: 216–272
- 10 Boneh D, Franklin M. Identity-based encryption from the Weil pairings. In: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology. London: Springer, 2001. 213–229
- 11 Cha J, Cheon J. An identity-based signature from gap Diffie-Hellman groups. In: Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography. London: Springer, 2003. 18–30
- 12 Joux A. A one round protocol for tripartite Diffie-Hellman. In: Proceedings of the 4th International Symposium on Algorithmic Number Theory. London: Springer, 2000. 385–394
- 13 Barreto P, Galbraith S, Heigearthaigh C, et al. Efficient pairing computation on supersingular Abelian varieties. *Design Code Cryptogr*, 2007, 42: 239–271
- 14 Beuchat J, Gonzalez J, Mitsunari S, et al. High-speed software implementation of the optimal ate pairing over Barreto-Naehrig curves. In: Proceedings of the 4th International Conference on Pairing-Based Cryptography. Berlin: Springer, 2010. 21–39
- 15 Hess F, Smart N, Vercauteren F. The Eta pairing revisited. *IEEE Trans Inf Theory*, 2006, 52: 4595–4602
- 16 Scott M, Costigan N, Abdulwahab W. Implementing cryptographic pairings on smartcards. In: Proceedings of the 8th International Conference on Cryptographic Hardware and Embedded Systems. Berlin: Springer, 2006. 134–147
- 17 Chevallier-Mames B, Coron J, McCullagh N, et al. Secure delegation of elliptic-curve pairing. In: Proceedings of the 9th IFIP WG 8.8/11.2 International Conference on Smart Card Research and Advanced Application. Berlin: Springer, 2010. 24–35
- 18 Galbraith S, Paterson K, Smart N. Pairings for cryptographers. *Discrete Appl Math*, 2008, 156: 3113–3121
- 19 Tsang P, Chow S, Smith S. Batch pairing delegation. In: Proceedings of the 2nd International Workshop on Security. Berlin: Springer, 2007. 74–90
- 20 Chow S, Au M, Susilo W. Server-aided signatures verification secure against collusion attack. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security. New York: ACM, 2011. 401–405
- 21 Chen X F, Susilo W, Li J, et al. Efficient algorithms for secure outsourcing of bilinear pairings. *Theor Comput Sci*, 2015, 562: 112–121
- 22 Tian H B, Zhang F G, Ren K. Secure bilinear pairing outsourcing made more efficient and flexible. In: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security. New York: ACM, 2015. 417–426
- 23 Boneh D, Franklin M. Identity-based encryption from the Weil pairings. In: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology. London: Springer, 2001. 213–229
- 24 Boneh D, Lynn B, Shacham H. Short signatures from the Weil pairings. In: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security. London: Springer, 2001. 514–532
- 25 Cha J, Cheon J. An identity-based signature from gap Diffie-Hellman groups. In: Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography. London: Springer, 2003. 18–30
- 26 Galbraith S, Paterson K, Smart N. Pairings for cryptographers. *Discrete Appl Math*, 2008, 156: 3113–3121
- 27 Sahai A, Waters B. Fuzzy identity-based encryption. In: Proceedings of 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2005. 457–473
- 28 Waters B. Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology. Berlin: Springer, 2009. 619–636
- 29 Boyen X, Waters B. Anonymous hierarchical identity-based encryption (without random oracles). In: Proceedings of the 26th Annual International Conference on Advances in Cryptology. Berlin: Springer, 2006. 290–307
- 30 Paterson K, Schuldt J. Efficient identity-based signatures secure in the standard model. In: Proceedings of the 11th Australasian Conference on Information Security and Privacy. Berlin: Springer, 2006. 207–222

New algorithms for verifiable outsourcing of bilinear pairings

Yanli REN^{1*}, Ning DING^{2,3}, Tianyin WANG⁴, Haining LU⁵ & Dawu GU²

¹ School of Communication and Information Engineering, Shanghai University, Shanghai 200444, China;

² School of Electronic Information and Electrical Engineering, Shanghai Jiaotong University, Shanghai 200240, China;

³ State Key Laboratory of Cryptology, Beijing 100878, China;

⁴ School of Mathematical Science, Luoyang Normal University, Luoyang 471022, China;

⁵ School of Information Security and Engineering, Shanghai Jiaotong University, Shanghai 200240, China

*E-mail: renyanli@shu.edu.cn

Abstract It is well known that the computation of bilinear pairing is the most expensive operation in pairing-based cryptography. In this paper, we first propose a secure verifiable outsourcing algorithm of single bilinear pairing based on two servers in the one-malicious model. The outsourcer need not execute any expensive operations, such as scalar multiplication or modular exponentiation. We also present another verifiable outsourcing algorithm for multiple bilinear pairings. This algorithm is much more efficient than the repeated invocation of the outsourcing algorithm of single bilinear pairing. Both of the proposed algorithms enable the outsourcer to detect any failure with a probability of 1 if one of the servers misbehaves. Finally, we utilize the proposed algorithms as two subroutines to achieve an anonymous identity-based encryption (AIBE) scheme with outsourced decryption and an identity-based signature (IBS) scheme with outsourced verification.

Keywords cloud computing, verifiable, outsourcing computation, bilinear pairing, modular exponentiation, identity-based encryption

Yanli REN was born in 1982. She is an associate professor in the School of Communication and Information Engineering at Shanghai University, China. She was awarded a M.S. degree in applied mathematics in 2005 from Shaanxi Normal University, China, and a Ph.D. degree in computer science and technology in 2009 from Shanghai Jiaotong University, China. Her research interests include applied cryptography, outsourcing computing, network security.

Ning DING was born in 1978. He is now an assistant researcher at the Department of Computer Science and Engineering of Shanghai Jiaotong University, China. He obtained his Bachelor's degree in computer science from Central South University, China in 2000 and his Ph.D. degree in computer science from Shanghai Jiao Tong University, China in 2009. His main research areas are cryptography and information security.

Tianyin WANG was born in 1979. He received the B.S. degree in mathematics from Henan University, in 2002, the M.S. degree in applied mathematics from Shaanxi Normal University, in 2005, and the Ph.D. degree in cryptography from Beijing University of Post and Telecommunications in 2010. He is currently an associate professor at Luoyang Normal University. His research interests include cryptography and in-

formation security.