Computer Science and Application 计算机科学与应用, 2024, 14(6), 32-40
Published Online June 2024 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2024.146139

Hans汉斯

基于TEE的多客户端函数加密方案研究

程钰雯，岳笑含

沈阳工业大学信息科学与工程学院，辽宁 沈阳

收稿日期：2024年5月14日；录用日期：2024年6月14日；发布日期：2024年6月21日

摘　要

随着云计算的兴起，对加密数据进行联合计算变的越来越重要。近年来，多客户端函数加密的发展使用户能够在不需要任何交互的情况下对私有输入进行联合计算。在云计算中，确保安全性与可靠性是极为重要的需求。针对多客户端函数加密方案出现的隐私安全问题，本文提出了一种基于TEE技术的去中心化多客户端函数加密方案。利用Python语言、开发环境PyCharm 2020.3.0版本、以及Python的PyCryptodome V3.10.1密码学库下对本文方案的关键算法或协议进行了仿真实验。综上试验结果及分析表明本方案在保障隐私安全的同时，具有较好的计算性能。这为在云计算环境中处理敏感数据提供了一种高效且安全的解决方案，因此具有一定的实际应用意义。

关键词

多客户端函数加密，去中心化，TEE，加权阈值签名，隐私保护

Research on Multi-Client Functional Encryption Scheme Based on TEE

Yuwen Cheng, Xiaohan Yue

School of Information Science and Engineering, Shenyang University of Technology, Shenyang Liaoning

Received: May 14th, 2024; accepted: Jun. 14th, 2024; published: Jun. 21st, 2024

Abstract

With the rise of cloud computing, joint computing of encrypted data is becoming increasingly important. In recent years, the development of multi-client function encryption has enabled users to perform joint computations on private inputs without any interaction. In cloud computing, ensuring security and reliability is an extremely important requirement. Aiming at the privacy and security problems of multi-client function encryption schemes, this paper proposes a decentralized multi-client function encryption scheme based on Intel SGX technology. Using Python language, de-

文章引用: 程钰雯, 岳笑含. 基于 TEE 的多客户端函数加密方案研究[J]. 计算机科学与应用, 2024, 14(6): 32-40.
DOI: 10.12677/csa.2024.146139

velopment environment PyCharm 2020.3.0 version, and Python PyCryptodome V3.10.1 cryptography library, the key algorithms or protocols of this scheme are simulated. In summary, the experimental results and analysis show that the scheme has good computational performance while protecting privacy and security. This provides an efficient and secure solution for processing sensitive data in the cloud computing environment, so it has certain practical application significance.

Keywords

Multi-Client Functional Encryption, Decentralized, TEE, Weighted Threshold Signature, Privacy Protection

1. 引言

加密是用户在不安全的网络或存储站点上安全地共享数据的一种方法。在公钥密码学[1]出现之前，一个被广泛接受的观点是，两个用户之间想要秘密的进行通信，他们需要先建立一个互相都持有的秘密密钥 k，即对称加密[2]。在传统的对称加密中，同一个密钥被用于加密和解密，这就要求通信双方在进行加密通信之前必须共享同一个密钥，这种方式对于一些小型组织来说是可以接受的，但是对于大型网络，比如由数十亿用户组成的 Internet 网络，显然是不可行的，因为密钥的分发和管理是一个复杂的问题。

函数加密(Functional Encryption, FE) [3]是一种新兴的加密范式，它以一种更加灵活的方式扩展了“All-or-Nothing”的公钥加密的要求，旨在解决传统加密方法无法提供的灵活性和功能性问题。函数加密的核心思想是允许对加密数据进行特定功能的计算，而不是只允许解密。然而，用户输入的数据可能很大，比如一个多维的向量，但函数加密的定义意味着输入的数据只能来自一方：即向量的所有坐标都是由一方提供的，并且是加密的。而在许多实际的应用场景中，数据可能是来自不同各方信息的集合，这些参与方之间可能互相并不信任。

多客户端函数加密(Multi-Client Functional Encryption, MCFE) [4]解决了前者独立生成密文的问题，采用去中心化的多客户端函数加密(Decentralized Multi-Client Functional Encryption, DMCFE) [5]来解决后者中心权威不可信时泄露客户端私有数据的问题。DMCFE 移除了中央权威机构或主密钥。

一个理想的 DMCFE 方案，不仅要保障数据的机密性和完整性，还要满足抗攻击性和可用性的特点。而可信执行环境(Trusted Execution Environment, TEE) [6]可以很好的达到机密性和可验证性的需求，解决现有函数加密系统中出现的隐私安全和可实现性的问题。通过在 TEE 中执行 DMCFE 算法，可以确保用户数据在计算过程中得到安全保护，提高了数据隐私性，也相应提升了加密算法的可用性和实用性。

2. 技术背景

2.1. 加权阈值签名

椭圆曲线数字签名算法(Elliptic Curve Digital Signature Algorithm, ECDSA)，广泛用于保护数字信息的完整性和身份验证[7]。与传统的离散对数问题和大整数分解问题相比，椭圆曲线密码学因其抵御亚指数算法攻击的特性而在公钥密码学领域独具优势。以下是对 ECDSA 签名算法的形式化定义：

系统建立：输入安全参数，输出算法的公开参数 $param=\{E,G,P,p,q,h\}$，其中，E 为定义在有限域 F_q 上的椭圆曲线，p 为一个素数，G 为椭圆曲线上所有整数点构成的加法群，P 和 q 分别为群 G 的生成元和素数阶，h 为输入映射到 Z_q^* 域上的杂凑函数，即 $h:\{0,1\}^* \to Z_q^*$。Z_q^* 域由整数集合 $\{1,2,\cdots,q-1\}$ 构成。

密钥生成：输入算法的公开参数 $param$，输出签名私钥 d 和验证公钥 Q，其中 $d \in Z_q^*$ 为随机秘密值，公钥 $Q=dP$ 公开可见。

签名生成：输入算法的公开参数 $param$、用户私钥 d 和待签名消息 m，输出签名 $\sigma=(r,s)$。步骤如下。

1) 选择一个随机数 $k \in Z_q^*$，计算 $R=k\cdot P=(r_x,r_y)$，其中 r_x 和 r_y 分别为椭圆曲线上点 R 的横坐标和纵坐标。

2) 计算 $r=r_x \bmod q$，若 $r=0$，则返回步骤(1)；否则，执行步骤(3)。

3) 计算 $s'=k^{-1}(e+dr)\bmod q$，其中，$e=h(m)$ 为消息摘要。

4) 输出签名信息 $\sigma=(r,s)$，其中 $s=\min\{s',q-s'\}$，$\min\{\}$ 函数表示取集合中较小的数值。

签名验证：输入待验证的消息 m 和签名 σ，输出验证结果“1”或“0”。具体步骤如下：

1) 解析签名 σ 获得 (r,s)，计算摘要 $e=h(m)$。

2) 计算 $R_v=s^{-1}(eP+rQ)=(x_v,y_v)$。

3) 若 $r=x_v \bmod q$，输出 1；否则输出 0。

2.2. 多客户端函数加密

MCFE 是一种密码学技术，旨在允许多个客户端合作执行加密计算，而不必暴露原始输入或中间结果[8]。这种技术通常涉及将计算任务分解成多个部分，然后由不同的客户端处理各自的部分，最终得到加密的结果。这样的方法有助于保护数据隐私，因为每个客户端只需处理其分配的部分，而无需了解整个计算的细节。MCEF 常用于安全多方计算(Secure Multi-Party Computation, SMPC) [9]，其中多个参与方合作执行计算，同时保持输入数据的隐私。

MCFE：一个在一组包含 n 个发送方的消息 M 上的 MCFE 由以下四个算法组成：

1) $\text{Setup}(\lambda)$：输入安全参数 λ，输出公共参数 mpk、主密钥 msk 和 n 个客户端的加密密钥 ek_i；

2) $\text{Encrypt}(ek_i,x_i,\ell)$：输入用户的加密密钥 ek_i，要加密的值 x_i，和一个标签 ℓ，输出密文 $C_{\ell,i}$；

3) $\text{DKeyGen}(msk,f)$：输入主密钥 msk，函数 $f:M^n \to R$，输出函数解密密钥 dk_f；

4) $\text{Decrypt}(dk_f,\ell,\boldsymbol{C})$：输入函数解密密钥 dk_f，标签 ℓ 和一个 n 维的密文 $\boldsymbol{C}$，如果 $\boldsymbol{C}$ 是关于标签 ℓ，$\boldsymbol{x}=(x_i)_i \in M^n$ 的有效加密，则输出 $f(\boldsymbol{x})$；否则输出 $\perp$。

正确性：假设 mpk 包含在 msk 中，同时也包含在所有的加密密钥 ek_i 以及函数解密密钥 dk_f 中。正确性属性表明，给出 $(mpk,msk,(ek_i)_i) \leftarrow \text{SetUp}(\lambda)$，对于任何标签 ℓ，任何函数 $f:M^n \to R$，和任一向量 $\boldsymbol{x}=(x_i)_i \in M^n$，如果 $C_{\ell,i} \leftarrow \text{Encrypt}(ek_i,x_i,\ell)$，$i \in \{1,\cdots,n\}$，函数解密密钥 $dk_f \leftarrow \text{DKeyGen}(msk,f)$，那么 $\text{Decrypt}(dk_f,\ell,\boldsymbol{C}_\ell=(C_{\ell,i})_i)=f(\boldsymbol{x}=(x_i)_i)$。

3. 提出的方案

3.1. 方案的模型和形式化定义

本方案定义的系统模型如图 1 所示，其中包含了两类参与的实体，分别是客户端应用程序(Client Application, CA)和解密节点云平台(Decryption Node Cloud Platform, DNCP)，其中解密节点云平台又包括解密飞地(Decryption Enclave, DE)和函数飞地(Function Enclave, FE)。各实体的具体职责如下：

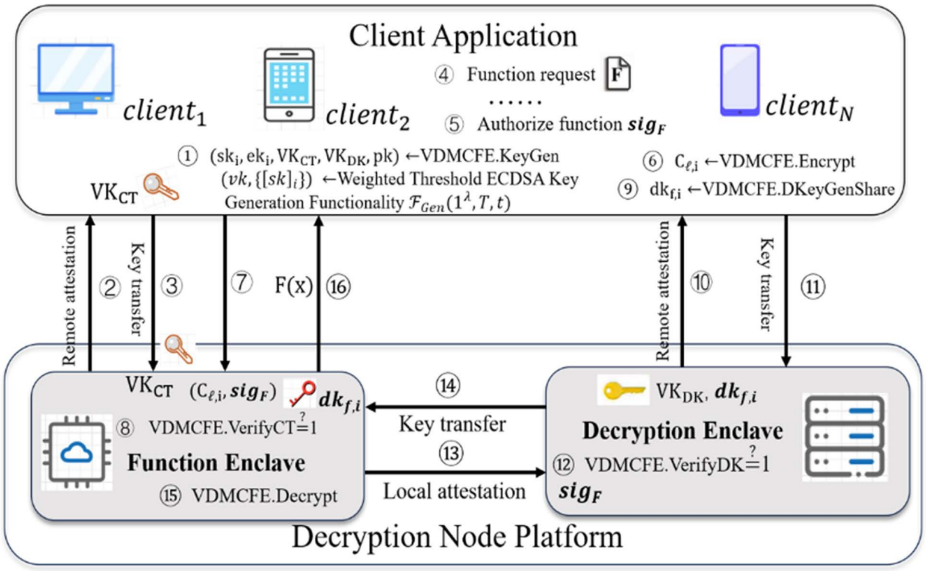

Figure 1. System model diagram
图 1. 系统模型图

1) 客户端平台(Client Application)：由于本方案是去中心化的，没有可信第三方用于生成函数解密密钥以及系统所需的参数，因此在本文方案中，客户端负责系统的初始化过程，生成 DMCFE 算法所需要的私钥、加密密钥、密文以及解密密钥的验证密钥和公开参数。此外，还负责生成加权阈值 ECDSA 签名算法的签名密钥和验证密钥，用于在函数授权阶段使用签名密钥对函数 F 执行授权。

2) 解密节点云平台(Decryption Node Platform)：

解密飞地(DE)：DE 与客户端平台通过远程认证的方式建立起一个安全信道，将在客户端生成的函数解密密钥通过安全信道传输到 DE 中，由于客户端生成的验证密钥是公开可见的，因此在 DE 中使用验证密钥对客户端生成的函数解密密钥执行验证。此外，对函数 F 的签名也在 DE 中使用签名算法生成的严密密钥进行验证。

函数飞地(FE)：由于 FE 与 DE 位于同一平台，因为 FE 与 DE 之间通过本地认证的方式建立起一个安全信道，DE 将验证通过的函数解密密钥通过安全信道传输到 FE 中，除此之外，客户端通过与 FE 远程认证建立起安全信道，将在本地加密的数据通过安全信道传输到 FE 中，由密文的验证密钥在 FE 中对密文进行验证，若验证通过，将在 FE 中对密文执行解密算法，最后生成关于明文信息 x 的函数值 $f(x)$ 发送给客户端。

3.2. 方案的具体构建

1) 系统初始化阶段

$\text{HW.Setup}(1^\lambda) \to params$：在系统初始化阶段，解密节点平台 DN 对 HW 安全硬件执行 $\text{HW.Setup}(1^\lambda)$ 并记录输出的参数 $params$。

$\text{Setup}(\lambda) \to pp$：在系统初始化阶段，由客户端平台运行初始化算法，生成全局公开参数 pp 以及客户端本地的公私钥对。各实体通过公共参数执行密钥生成算法生成各自的公私钥对。

2) 密钥生成阶段

$\mathcal{F}_{Gen}(1^\lambda, T, t) \to (vk, \{[sk]_i\})$：本文的加权阈值 ECDSA 签名方案从将秘密签名密钥 sk 在所有参与方之间进行加权随机秘密共享(WRSS)开始。

VDMCFE.KeyGen$(\lambda)\to(sk_i, ek_i, vk_{CT}, vk_{DK}, pk)$：VDMCFE 的密钥生成输入安全参数 λ，输出私钥 sk_i、加密密钥 ek_i、密文的验证密钥 vk_{CT}，函数解密密钥的验证密钥 vk_{DK}，以及公钥 pk。用于后续对数据的加密、函数解密密钥的生成，密文及密钥的验证等。

3) 函数授权阶段

加权阈值 ECDSA 签名算法将签名协议分为两个阶段：一个预签名协议，仅依赖于签名密钥的份额，另一个是依赖于实际消息的非交互式签名协议。客户端发出函数请求 F，对 F 执行签名。

4) 加密阶段

VDMCFE.Encrypt$(ek_i, x_i, \ell, m)\to C_{\ell,i}$：客户端在本地执行加密算法，利用加密密钥 ek_i 对本地数据进行加密，客户端与 FE 通过远程认证的方式建立安全信道，生成的密文通过安全信道传输到 FE 中。

5) 解密密钥的生成及验证阶段

VDMCFE.DKeyGenShare$(sk_i, \ell_y, m, pk)\to dk_{i,y}$：客户端在本地执行函数解密密钥生成算法，利用私钥 sk_i 生成函数解密密钥 $dk_{i,y}$，与 DE 通过远程认证建立起安全信道，客户端将函数解密密钥 $dk_{i,y}$ 发送到 DE，在 DE 内执行密钥的验证算法。

6) 密文的验证及解密阶段

VDMCFE.Verify$\left(\left(C_{\ell,i}\right)_{i\in[n]}, vk_{CT}, m\right)$：客户端对本地数据加密生成的密文，通过与 FE 远程认证建立的安全信道，将 $C_{\ell,i}$ 传输到 FE。由 vk_{CT} 在 FE 内对 $C_{\ell,i}$ 执行验证，验证通过后，DE 与 FE 通过本地认证的方式建立起安全信道，DE 将解密密钥 $dk_{f,i}$ 传输到 FE，在 FE 内对密文解密，最后解密结果值 $F(x)$ 发送给客户端。

4. 实验

4.1. 仿真设计

本方案运行在 Intel Skylake i7-6700 处理器，主频为 3.40 GHz，内存为 8 GiB 的 RAM，安装有 Windows Server 2012 R2 标准操作系统的平台上测试了原型实现。编程语言和版本 Python 3.8、开发环境 PyCharm 2020.3.0 版本、以及 Python 的 PyCryptodome V3.10.1 密码学库下对本文的方案进行了仿真实现，并使用了 Intel (R) SGX SDK 1.6 和 Intel (R) SGX PSW 1.6 附加组件，给出了核心算法或协议的代码及运行结果。客户端仿真示意图如图 2 所示。

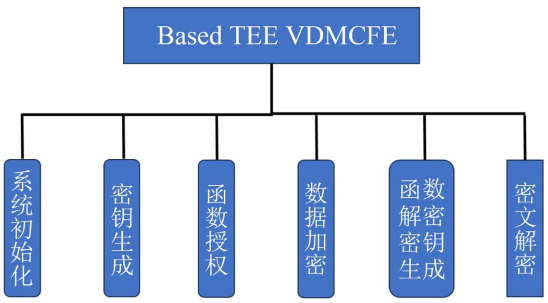

Figure 2. Client simulation diagram
图 2. 客户端仿真示意图

在仿真实验中首先定义了 RSA 公钥加密类[10]、RSA 签名类[11]及 ECDSA 签名类[12]。

1) 系统初始化阶段：客户端执行系统初始化 Setup 算法，生成全局公共参数，以及各个实体根据公共参数输出各自的公私钥对。如下是客户端执行初始化算法的核心代码，其执行结果如图 3 所示。

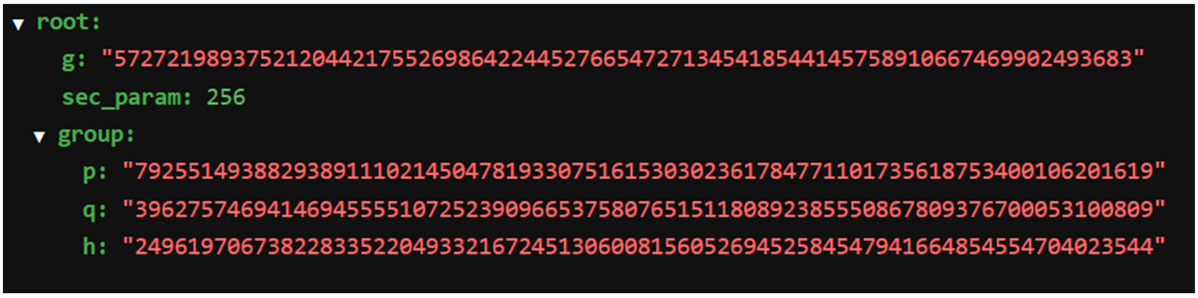

Figure 3. Result graph of client execution initialization algorithm
图 3. 客户端执行初始化算法结果图

2) 签名密钥的生成阶段：客户端执行加权阈值 ECDSA 签名算法，生成各自的签名密钥和验证密钥。如下是客户端执行 ECDSA 签名算法的密钥生成阶段核心代码，其执行结果如图 4 所示。

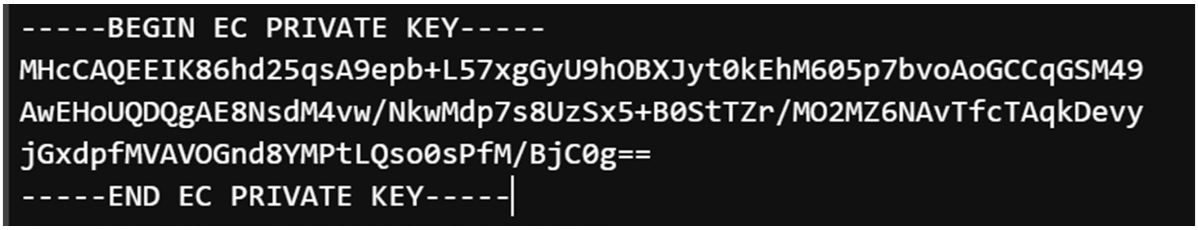

Figure 4. Result graph of ECDSA key generation algorithm executed by the client
图 4. 客户端执行 ECDSA 密钥生成算法结果图

3) 函数授权阶段：用户发送函数请求 F，客户端使用各自的签名密钥对函数 F 执行签名授权过程。如下是客户端执行 ECDSA 签名算法签名阶段的核心代码，其执行结果如图 5 所示。

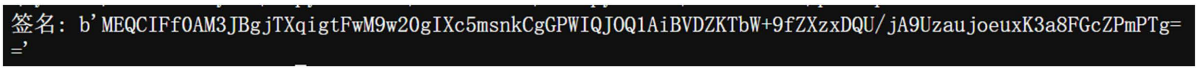

Figure 5. Result graph of ECDSA signature algorithm executed by the client
图 5. 客户端执行 ECDSA 签名算法结果图

4) 客户端加密阶段：客户端使用加密密钥对本地数据进行加密，生成密文 ct 发送到函数飞地。如下是客户端执行加密算法的核心代码，其执行结果如图 6 所示。

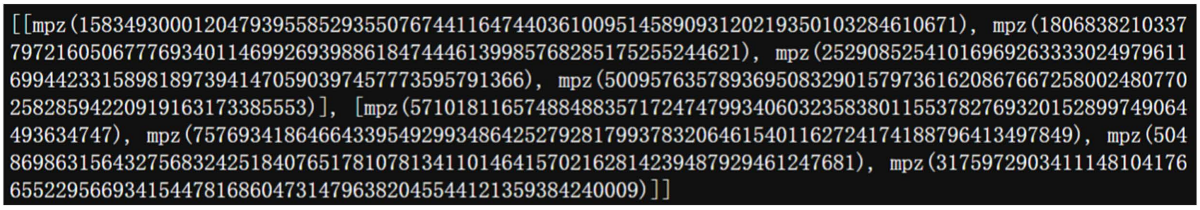

Figure 6. Result graph of client executing encryption algorithm
图 6. 客户端执行加密算法的结果图

5) 函数解密密钥生成及验证阶段：客户端使用自己的私钥生成部分函数解密密钥，并将函数解密密钥通过安全信道传输到解密飞地，在解密飞地内执行验证。如下是客户端执行函数解密密钥生成算法的核心代码，其执行结果如图 7 所示。

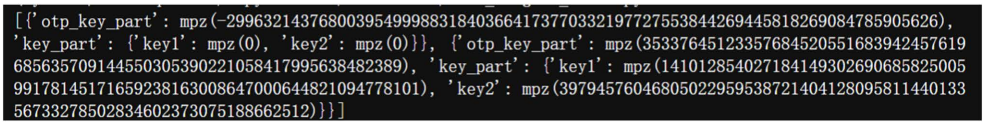

Figure 7. Result graph of decryption key generation algorithm for client execution function
图 7. 客户端执行函数解密密钥生成算法的结果图

6) 密文解密阶段：函数解密密钥通过本地认证传输到函数飞地，在函数飞地内对密文执行解密，最后生成的函数值 $f(x)$发送给客户端。如下是函数飞地执行密文解密算法的核心代码，其执行结果如图 8 所示。

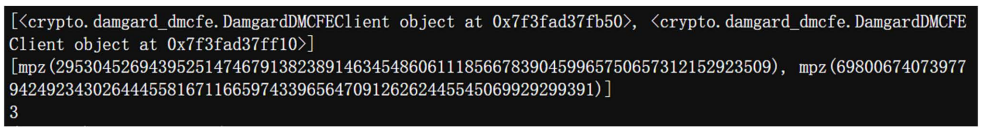

Figure 8. The result graph of the function enclave executing the ciphertext decryption algorithm
图 8. 函数飞地执行密文解密算法的结果图

4.2. 性能测试

本文设计的方案计算了 VDMCFE-HW.Setup，VDMCFE-HW.Keygen，VDMCFE-HW.Encryption，VDMCFE-HW.VerifyCT，VDMCFE-HW.DKeyGenShare，VDMCFE-HW.VerifyDK，VDMCFE-HW.DKeyComb 和 VDMCFE-HW.Decrypt 算法的平均时间和标准差。表 1 包含了 VDMCFE.Setup 和 VDMCFE.KeyGen 算法运行的时间。

Table 1. VDMCFE algorithm runtime
表 1. VDMCFE 算法运行时间

创建飞地	52 ms
VDMCFE.Setup	2.1675 ms
VDMCFE.KeyGen	2.1781 ms
VDMCFE.Encrypt	2.1889 ms
Total	58.5345 ms

本方案评估了三种函数加解密算法的性能，分别是基于身份的加密(IBE)、order revealing encryption (ORE)和三输入 DNF (3DNF)。为了展示原语的 SGX 辅助版本在性能上与纯加密版本(如配对的 IBE、DNF 和多线性映射 3DNF)的比较，本文选择这些函数进行研究。表 2 总结了三种函数的解密时间，包括解密过程三个主要 ecalls 所花费时间的细分：创建飞地、DE 本地认证、解密密文和函数评估。

Table 2. Time comparison of IBE, ORE, and 3DNF algorithms
表 2. IBE，ORE 和 3DNF 算法时间对比

Functionality	IBE [13]	ORE [14]	3DNF [15]
create enclave	13 ms	18.9 ms	17.6 ms
local attest	1.4 ms	1.6 ms	1.5 ms
decrypt & eval	0.89 ms	0.71 ms	0.84 ms
Total	15.29 ms	21.21 ms	19.94 ms

如表 2 所示，对于每个函数来说，创建 enclave 所需的时间比解密和评估它所需的时间长两个数量级。一旦函数 enclave 被创建并且对 DE 的本地认证完成，同一个 enclave 可以用来解密任意数量的输入密文元组。因此，对许多密文或密文元组进行解密的摊销成本远低于对单个输入进行解密的成本。在下图 9 中显示了对包含 1000 个输入的密文元组进行解密的摊销成本。

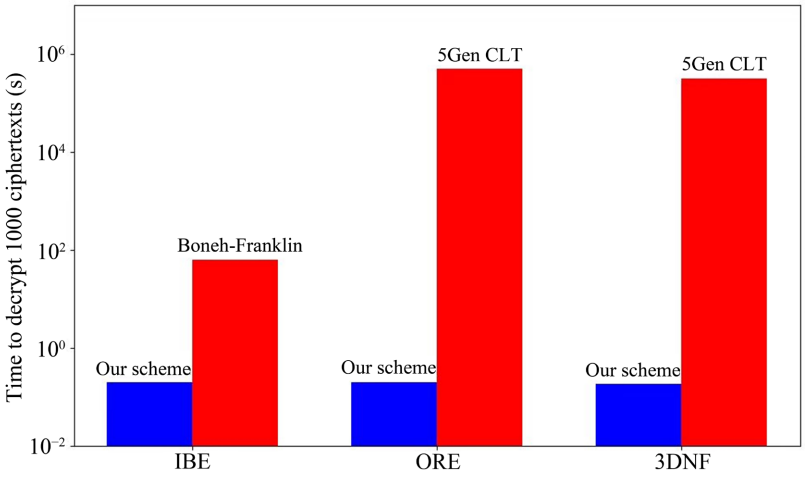

Figure 9. The average cost of decrypting 1000 ciphertext tuples
图 9. 输入 1000 个密文元组运行解密的平均代价

实验数据表明，通过在计划平台上对 IBE 的解密时间进行测量，可获得关键结果。本研究还包含提出了基于 ORE 和 3DNF 相关的 5Gen 解密时间性能指标。本文认为无需在本方案平台上对 ORE 和 3DNF 的 5Gen 实施进行评估，因为它们相较于本方案基于 SGX 的实施而言性能慢了四个数量级。本系统成功地实现了多输入函数，使得原语的构建成为了可能。在没有安全硬件的情况下，这些原语此前无法被用于实际应用，本文方案填补了这一空白。

5. 结论

本文通过对目前函数加密方案中存在的安全及信任问题深入分析，并在可信执行环境的基础上提出了基于 SGX 的去中心化多客户端函数加密方案。本文通过引入 DMCFE、加权阈值签名算法与上述 SGX 机制相结合来设计并实现安全方案。该方案旨在使用户代码及数据在分布式计算过程中始终保持加密状态，只有在可信执行环境中才解密执行，从而达到不向任何参与方透露信息的目的。最后基于 Python 语言、开发环境 PyCharm 2020.3.0、以及 Python 的 PyCryptodome V3.10.1 密码学库对本文的方案进行了仿真实验。最后从计算代价、通信代价上对本方案关键算法的性能进行了对比分析。实验结果表明，本文提出的安全方案可以保证用户数据在分布式计算中的安全性，基本实现了预期目标。

参考文献

[1] Diffie, W. and Hellman, M.E. (2022) New Directions in Cryptography. In: Slayton, R., Ed., *Democratizing Cryptography*: *The Work of Whitfield Diffie and Martin Hellman*, Association for Computing Machinery, 365-390.

[2] 陈立全, 张林樾, 陈垚. 一种结合可验证数据库的属性可搜索加密方案[J]. 密码学报, 2022, 9(5): 910-922.

[3] Kim, S., Lewi, K., Mandal, A., *et al*. (2018) Function-Hiding Inner Product Encryption Is Practical. In: *International Conference on Security and Cryptography for Networks*, Springer International Publishing, 544-562.

[4] 张志强, 朱友文, 王箭, 等. 基于内积谓词的属性基隐私保护加密方案[J]. 电子与信息学报, 2023, 45(3): 828-835.

[5] Mehibel, N. and Hamadouche, M. (2019) A New Enhancement of Elliptic Curve Digital Signature Algorithm. *Journal of Discrete Mathematical Sciences and Cryptography*, **23**, 743-757. https://doi.org/10.1080/09720529.2019.1615673

[6] Chandramouli, A., Choudhury, A. and Patra, A. (2022) A Survey on Perfectly Secure Verifiable Secret-Sharing. *ACM Computing Surveys*, **54**, 1-36. https://doi.org/10.1145/3512344

[7] 钱文君, 沈晴霓, 吴鹏飞, 等. 大数据计算环境下的隐私保护技术研究进展[J]. 计算机学报, 2022, 45(4): 669-701.

[8] Felsen, S., Kiss, Á., Schneider, T. and Weinert, C. (2019) Secure and Private Function Evaluation with Intel SGX. *Proceedings of the* 2019 *ACM SIGSAC Conference on Cloud Computing Security Workshop*, London, 11 November 2019, 165-181. https://doi.org/10.1145/3338466.3358919

[9] Zheng, W., Wu, Y., Wu, X., Feng, C., Sui, Y., Luo, X., *et al*. (2020) A Survey of Intel SGX and Its Applications. *Frontiers of Computer Science*, **15**, Article ID: 153808. https://doi.org/10.1007/s11704-019-9096-y

[10] Fisch, B., Vinayagamurthy, D., Boneh, D., *et al*. (2017) Iron: Functional Encryption Using Intel SGX. *Proceedings of the* 2017 *ACM SIGSAC Conference on Computer and Communications Security*, Dallas, 30 October-3 November 2017, 765-782.

[11] Abdalla, M., Bourse, F., De Caro, A., *et al*. (2015) Simple Functional Encryption Schemes for Inner Products. In: *IACR International Workshop on Public Key Cryptography*, Springer, 733-751.

[12] Mera, J.M.B., Karmakar, A., Marc, T., *et al*. (2022) Efficient Lattice-Based Inner-Product Functional Encryption. In: *IACR International Conference on Public-Key Cryptography*, Springer International Publishing, 163-193.

[13] Agrawal, S., Libert, B., Maitra, M., *et al*. (2020) Adaptive Simulation Security for Inner Product Functional Encryption. In: *IACR International Conference on Public-Key Cryptography*, Springer International Publishing, 34-64.

[14] Shi, E., Chan, H.T.H., Rieffel, E., *et al*. (2011) Privacy-Preserving Aggregation of Time-Series Data. In: *Annual Network & Distributed System Security Symposium* (*NDSS*), Internet Society, 489-505.

[15] Benhamouda, F., Joye, M. and Libert, B. (2016) A New Framework for Privacy-Preserving Aggregation of Time-Series Data. *ACM Transactions on Information and System Security*, **18**, 1-21. https://doi.org/10.1145/2873069