Fidelius: 基于 Intel SGX 和区块链的新型安全数据分析框架

张金波

July 30, 2025

Abstract

随着大数据时代对数据分析的依赖日益增长，数据泄露和隐私侵犯的担忧变得越来越普遍。虽然现有的技术如安全多方计算（MPC）、同态加密（HE）、联邦学习（FL）和可信执行环境（TEE）旨在解决这些问题，但在处理涉及多个角色的复杂数据分析场景时，它们往往表现出局限性。在本文中，我们提出了 Fidelius，一个利用 Intel SGX 和区块链来增强数据分析安全性的新型系统。Fidelius 采用静态二进制分析方法隐私描述语言（PDL）来防止计算结果中的数据泄露。此外，它引入了一种密码协议来确保计算结果的可靠性和可验证性，并结合密码协议和本地认证来实现分析程序的一致性验证。实验结果表明，Fidelius 在性能方面超越了现有解决方案，同时产生的开销最小。因此，Fidelius 为增强涉及多个角色的复杂场景中的数据分析安全性提供了一个有前景的解决方案。

1 引言

在大数据的时代，人们越来越依赖于数据分析进行决策。例如，当一家超市想要合理进货以便最大化其利润时，他需要考虑超市历史的售卖订单、市场上客户对于商品的喜爱度、各类商品的利润率等方面，这往往需要通过分析各个方面大量的数据来做决定。

为了完成数据分析任务，那些没有服务器的人通常把数据分析的任务托管到云服务器上。但在使用云服务器进行数据分析的同时，人们会担心这些重要的数据遭到泄漏，因为这个数据往往涉及到个人的隐私、商业机密等。

为了避免数据的泄漏的问题，越来越多的相关技术被提出以解决数据泄露的问题。这些技术包括：多方安全计算、同态加密、联邦学习、可信执行环境等。

在这些技术中，MPC 和 HE 具备极高的安全性，因为它是密码学安全的，但由于 MPC 和 HE 包含了复杂的密码学操作，其性能相较于其他几类技术表现极差。FL 常见于机器学习的场景，主要用于联合建模，对于一般的数据分析场景并不适用。TEE 满足各类通用计算的场景，其计算性能较高，但其基于硬件的安全性，相较于 MPC 和 HE 的密码学安全，其安全性更低，而且经常出现新发现的基于 TEE 的侧信道攻击。

然而，随着数据分析的场景越来越复杂，现有的这些技术不能直接应用于那些复杂的场景，来解决数据泄漏的问题。

现有的数据分析场景中，仅存在租户和云计算服务提供商两个角色，现有技术主要是为了防止云计算服务提供商泄漏租户的数据。但是现在的数据分析场景中，包含了数据提供方、数据使用方、模型提供方、云计算服务提供商这些角色，数据需要防止被数据提供方之外的所有角色泄漏。除此之外，还需要维护数据分析场景中各方的权益，例如，数据使用方得到合理的分析结果，数据提供方、模型提供方、云计算服务提供商获得相应的收益。

因此，在针对复杂的数据分析场景时，我们需要检测数据使用方是否通过输出的计算结果中包含敏感信息以泄漏数据，模型提供方是否通过恶意的模型泄漏数据，云计算服务提供商通过篡改 hypervisor、操作系统、内存、磁盘等泄漏数据，数据提供方、模型提供方以及云计算服务提供商共同生成的计算结果是否合法。

很多相关的研究被提出，来解决复杂的数据分析场景时所面临的问题，包括 SDTE，PrivacyGuard 等。SDTE 在数据交易行业引入了一种创新的” 数据处理即服务” 模型。利用这个模型，它通过一系列交易协议实现安全的数据交易。PrivacyGuard 利用智能合约定义数据使用策略，并利用 TEE 进行高效的合约执行，同时保护私有数据。SPDS 与 PrivacyGuard 类似，使用智能合约定义数据使用策略，并实现两阶段交付协议以确保计算结果和支付的安全发布。

Amanuensis 探索了区块链技术和 TEE 的交集，以解决数据溯源、机密性和用户隐私在数据共享中的挑战。

然而，这些相关的研究在解决复杂的数据分析场景时，仍存在一些限制。

首先，这些相关的研究工作不能保证数据分析结果中不会泄漏数据。虽然这些工作引入了数据使用规则，用以描述哪些人以什么样的价格访问哪些类型的数据，但对数据进行分析的程序是存在作恶的可能的，分析程序的最终输出结果可能是包含敏感信息泄漏的。一个最简单的例子是，分析程序直接输出原始数据，这样数据分析结果就获取到了完整的原始数据。

一个 straightforward 的解决方案是，要求分析程序公开，数据提供方可以审核分析程序是否存在泄漏数据的代码。但是对于模型提供方而言，模型也是他们重要的资产，他们不希望模型公开。另一方面，对于复杂的分析程序来说，审核分析程序存在巨大的工作量，不可能要求每一个分析任务都去进行审核，这会大大降低数据分析的效率。

其次，当前的工作中并不能保证计算结果的可信与可验证。具体来说，数据使用方得到计算结果后，他不能确定这个计算结果确实是通过指定的数据与指定的模型运行之后得到的计算结果，而不是一个随机生成的结果。更进一步，数据使用方并没有任何方式去验证这个结果的正确性。如果计算结果的可信和可验证都无法保证，那么完全可以用一个伪造的计算结果来欺骗数据使用方，数据使用方的利益将会受到损害。

最后，当前的工作中每一次进行数据分析任务都需要使用 remote attestation 来保证分析程序的一致性，但是有相关工作表明，remote attestation 具有低效、依赖可信第三方的缺点。以 Intel SGX 中的 Enhanced Privacy ID（EPID）为例，一次 remote attestation 的过程会涉及到 Intel Provisioning Service（IPS）、Intel Attestation（IAS）Service、Intel-signed provisioning enclave（PvE）、Intel-signed quoting enclave（QE）以及分析程序 Enclave 之间的交互。这些服务和 Enclaves 之间的交互需要通过广域网传输数据，同时，传输的数据需要加密保证其安全性，对于频繁的数据分析任务而言，性能上会有较大的损失。另一方面，IPS 和 IAS 是 Intel 提供的中心化服务，remote attestation 需要依赖其稳定运行。

在本文中，我们提出了 Fidelius，一个利用 Intel SGX 和区块链来增强数据分析安全性的系统，解决了前面提到的局限性。其中，Intel SGX 保证数据分析过程的 security 和 integrity，区块链用于可信的传输、存储和验证。

为了解决计算结果泄漏数据的问题，Fidelius 使用了静态二进制分析的方法，检查模型提供方的分析程序是否遵循隐私描述语言。其中，我们引入的隐私描述语言将数据的运算规则描述为有限状态机，反映了从输入数据到输出数据的状态转换，不在隐私描述语言描述的状态转换，都会被认为违反了隐私规则。

为了解决计算结果的可信与可验证的问题，Fidelius 提供了一套密码协议，该协议中由数据使用方提供一个私钥，该私钥通过加密转发至分析程序的 Enclave 中，并签名计算结果。由于该私钥仅在指定的分析程序 Enclave 中解密获得，数据提供方、模型提供方、云计算服务提供商均无法获取，故只要被签名的计算结果能够通过验证，说明计算结果确实出自指定的分析程序且可验证。

为了解决 remote attestation 低效、持续依赖中心化服务的问题，Fidelius 结合设计的密码协议以及 local attestation 实现分析程序的一致性验证。Fidelius 设计了密钥管理的 Enclave，在初始化的过程中获取密钥的授权，在此后的所有数据分析任务中，使用该经授权的密钥执行 local attestation 完成分析程序的一致性验证。

本文的主要贡献总结如下：

- 首先，我们引入了隐私描述语言（PDL）结合静态二进制分析，严格强制执行数据机密性，防止计算结果中的敏感数据泄露。
- 其次，我们设计了一种密码协议来确保计算结果的可靠性和可验证性。
- 我们集成了密码协议与本地认证机制，确保在受保护环境中执行的分析程序的完整性和正确性。
- 最后，我们评估了 Fidelius 的性能，实验结果表明它产生的开销最小，对数据分析系统的贡献不到 2%，同时性能超越现有解决方案 30 倍以上。

2 背景

2.1 Intel SGX

Intel Software Guard Extensions (SGX) 是一个硬件级别的可信执行环境，用于保护程序执行。它建立了一个安全的私有内存区域，称为 Enclave Page Cache (EPC)，保护其内容免受外部进程的未授权访问或修改。这个指定的内存空间被认为是可信的，而超出它的任何区域都被视为不可信的，严格限制从不可信环境的访问。

部署在 EPC 内的程序被称为 enclave 程序。Enclaves 被设计为与不可信环境隔离运行，只能通过明确声明的方法进行交互，这些方法被称为 ECALL 或 OCALL 方法。

我们将介绍封装在 enclave 内的关键方法和操作。

- *sgx_get_key*(): 此函数检索基于当前 enclave 哈希值、CPU ID 和其他配置文件生成的对称密钥。每个 enclave 都有一个以这种方式生成的唯一对称密钥。除非声明特定的导出（OCALL）方法，否则对称密钥在 enclave 外部无法访问。
- *sgx_ecc*256*_create_key_pair*(): 此函数通过利用 SGX 的自包含随机数生成器生成 ECC-256 非对称密钥对。

为了安全地将私有消息从 enclave 存储到本地存储中，"seal" 操作使用 enclave 的对称密钥加密消息，然后本地存储。这种方法确保密封消息的明文对用户不可访问，因为对称密钥保持未公开。为了解封消息，用户使用指定的（ECALL）方法将消息传输到 enclave，enclave 使用自己的对称密钥解密消息。

2.2 远程认证

远程认证是 Intel SGX 提供的一种技术，用于证明在未知平台上运行的 enclave 程序的完整性。目前，Intel SGX 提供两种类型的远程认证：Intel Enhanced Privacy ID (Intel EPID) Attestation 和 Elliptic Curve Digital Signature Algorithm (ECDSA) Attestation。

基于 EPID 的远程认证通常包括以下步骤。首先，应用程序 enclave 向同平台的 quoting enclave 发起请求，生成应用程序 enclave 的 quote。其中，quoting enclave 是 Intel SGX 官方提供的 enclave。这两个 enclave 之间通过 local attestation 进行请求发送、quote 传输，其中，local attestation 仅为同一平台的两个 enclave 之间建立可信通道并进行数据传输。应用程序 enclave 在接收到 quote 之后，向远端的 Intel Attestation Service 发送，进行验证，得到验证的结果。

EPID 存在着诸多限制，例如，要求应用程序 enclave 所在的网络能够连接上 Intel Attestation Service，要求 Intel Attestation Service 服务在任何时候都高可用（即不存在单点故障或者下线）。

基于 ECDSA 的认证被引入以解决与 EPID 相关的限制。通过 Intel SGX Data Center Attestation Primitives (DCAP)，基于 ECDSA 的认证使提供商能够建立并提供第三方认证服务，消除了对 Intel 提供的远程认证服务的依赖。

Intel SGX DCAP 允许第三方提供 quoting enclave，并生成应用程序 enclave 的 quote。其中，第三方 quoting enclave 使用的 attestation key 可由 data center 内部生成。attestation key 的公钥将被发送至 Intel's Provisioning Certification Enclave (PCE) 并被签名/授权。在生成 quote 之后，quote 会发送给 data center 内部的 attestation service 进行验证。

SGX 提供两种类型的 enclave 间通信：本地认证（LA）和远程认证（RA）。本地认证仅限于同一平台上的两个 enclave，而远程认证促进跨平台通信。在本地认证期间，每个 enclave 可以验证交互 enclave 的哈希值或签名者。相比之下，远程认证需要可信的第三方，如 Intel 服务器或 DCAP，来验证此信息。

2.3 区块链

2.3.1 公链和联盟链

区块链分为公链和联盟链，公链所有人都可以访问，一个节点可以任意加入或者离开，没有任何权限控制。由于公链限制少，达成所有节点的共识就相对困难，导致其系统吞吐较低。公链

最开始用于跨境支付，后发展成为去中心化金融的底层技术框架。联盟链顾名思义，由多个联盟成员组成，联盟成员一般为企业、机构、政府单位等。联盟链作为联盟成员中的可信第三方，具备公开、透明的特点，主要用于数据的公开存证、溯源、验证等。由于联盟链中的成员数量不多，可通过高效的共识算法使其达成一致，因此联盟链具备较高的吞吐性能。

在 Fidelius 中，联盟链的利用作为可靠且容错的第三方实体，用于数据传输、数据存储等任务。相比之下，传统中介容易受到单点故障、数据篡改风险以及建立 P2P 私有网络的高成本影响。

2.3.2 智能合约

区块链的智能合约是链上程序，以其代码和执行过程的透明度而闻名。在 Fidelius 中，智能合约在验证云服务提供商提供的分析结果签名方面发挥着关键作用。一旦验证成功完成，数据分析的成功就得到保证，为所有相关方建立信任。

3 威胁模型与设计选择

在本节中，我们首先介绍系统中的角色，然后详细说明威胁模型，突出潜在风险并讨论各种类型的攻击。此外，我们建立关键假设，这些假设构成了 Fidelius 设计和实施的基础。随后，我们深入探讨为 Fidelius 做出的设计选择，旨在缓解已识别的风险并应对系统上的潜在攻击。

3.1 角色

- 数据提供方（DP）。数据提供方作为原始数据的唯一所有者，最初在区块链上发布元数据，包括哈希值和原始数据的必要描述。此元数据的准确性通过数据提供方的可信度进行验证，以成功数据分析的历史为例。
- 模型提供方（MP）。模型提供方为特定类型的数据提供分析程序。
- 云服务提供方（CSP）。云计算提供方提供数据分析所需的计算资源，并额外提供可信执行环境以确保安全的数据分析。在接收到数据分析任务请求后，云计算提供方有义务执行分析程序并将结果返回给区块链。
- 数据使用方（DU）。数据使用方通过检查区块链上的元数据选择所需的原始数据，并启动数据分析以从指定的分析程序获得结果。
- 区块链。区块链作为数据传输和存储的可靠、抗故障的第三方。具体而言，智能合约验证分析结果上签名的正确性。

3.2 威胁模型

系统包含数据提供方、模型提供方、云服务提供方以及数据使用方四个角色，四个角色之间是相互不信任的。

在数据分析的过程中，存在以下攻击：

- 数据窃取攻击：云服务提供方可能通过他们提供的硬件或软件资源窃取数据；数据使用方可能登录服务器窃取数据。模型提供方提供的程序可能包含旨在窃取原始或中间数据的恶意代码。
- 数据伪造攻击：数据提供方提供的数据与声称的不一致。
- 数据滥用攻击：数据提供方的数据在没有适当授权的情况下被用于不同的模型和云服务器。
- 结果伪造攻击：数据分析的结果并不准确反映模型的真实执行。

- 结果窃取攻击：数据分析的结果被数据提供方、模型提供方、云服务提供方或未知的攻击者非法获取。

我们在本文中不解决 Intel SGX 的侧信道攻击。我们假设 Intel SGX 的硬件功能如广告所示，确保 enclave 内的代码保持不变，内部变量的值受到保护，免受直接内存访问。值得注意的是，我们对 Intel 的依赖仅限于设置过程中的一次性交互；不需要与 Intel 服务器或 DCAP 进行进一步联系。相比之下，以前的方法依赖远程认证，要求 Intel 服务器或 DCAP 在每个数据分析任务中持续保持完整性和可用性。

我们假设存在抗碰撞哈希函数和安全签名及加密方案。具体而言，我们的签名包含 $nonce$，确保有效的签名对 $(msg, signed_{msg})$ 不能由没有私钥的对手生成，使历史签名无效。

3.3 设计选择

为了应对已识别的攻击，我们做出了以下设计选择。

- 抵御数据窃取攻击：所有存在于非可信环境（例如网络传输、云服务器的非 TEE 部分）的数据都经过加密处理，以防止数据使用方或云服务提供商直接窃取数据。模型提供方提供的分析程序，在进行数据分析前进行代码静态分析，防止模型提供方通过数据分析结果窃取数据。
- 抵御数据伪造攻击：执行数据分析前，检查数据哈希与声称的哈希是否一致。
- 抵御数据滥用攻击：引入数字签名技术，数据提供方用私钥签名使用数据的平台、模型以及模型的参数，在执行数据分析前，会验证该签名的有效性。
- 抵御结果伪造攻击：将数据使用方的私钥加密地传输到云服务器的可信执行环境中，并用这个私钥签名结果。当数据使用方拿到结果时，也会得到该私钥对结果的签名，若签名验证通过，说明结果确实是由可信执行环境中得到的，故结果未经伪造。这一思想类似于零知识证明。
- 抵御结果窃取攻击：数据分析结果由数据使用方的公钥加密，只能被数据使用方查看。

4 设计

4.1 概述

本节简要概述 Fidelius 中使用的 enclaves 和符号，以及其架构和工作流程。

4.1.1 Enclaves 和符号

在本文中，我们定义以下 enclaves 和符号：

- EKeyMgr。Enclave 密钥管理器管理非对称密钥，处理创建、删除并提供基本密码学功能，包括消息加密、解密、签名和签名验证。
- EAnalyzer。Enclave 分析器，一个 Intel SGX Enclave 格式的分析程序，确保程序在执行期间保持不变。
- (PK_{DP}, SK_{DP})，数据提供方的非对称密钥对。
- (PK_{DU}, SK_{DU})，数据使用方的非对称密钥对。
- (PK_{CSP}, SK_{CSP}) 表示云服务提供方的非对称密钥对，由 EKeyMgr 生成。私钥 SK_{CSP} 安全地存储在 enclave 内，确保云服务提供商无法提取。
- $H(\cdot)$ 表示哈希函数。

- $Enc(PK, msg)$ 表示使用 PK 加密 msg。
- $Dec(SK, cipher)$ 表示使用 SK 解密 $cipher$。
- $Sign(SK, msg)$ 表示使用 SK 对 msg 进行签名。
- $Verf(PK, msg, sig)$ 表示使用 PK 和 msg 验证 sig。
- $\mathcal{F}(SK, PK_{CSP}, H_{EA})$ 表示使用 PK_{CSP} 将 SK 转发到 EKeyMgr 的过程。在此上下文中，SK 在 EAnalyzer 中使用，由 H_{EA} 标识。函数 $\mathcal{F}(\cdot)$ 涉及 $Sign(SK, concat(PK_{CSP}, H_{EA}))$ 和 $Enc(PK_{CSP}, SK)$。

4.1.2 架构和工作流程

Fidelius 架构和数据分析工作流程如图所示。该过程从设置阶段开始，其中 CSP 中的 EKeyMgr 生成非对称密钥对。公钥 PK_{CSP} 通过 Intel 的远程认证服务进行验证，而私钥 SK_{CSP} 安全地存储在 EKeyMgr 内。同时，数据提供方（DP）和数据使用方（DU）各自生成自己的非对称密钥对，并保留这些密钥。此外，DP 使用她的公钥 PK_{DP} 加密数据，并为 DU 的使用准备这些加密数据。

DP 将加密数据上传到 CSP，并使用 $\mathcal{F}(SK_{DP}, PK_{CSP}, H_{EA})$ 将她的私钥 SK_{DP} 转发给 EKeyMgr。类似地，DU 将分析程序上传到 CSP，并使用相同的方法将她的私钥 SK_{DU} 转发给 EKeyMgr。重要的是要注意，明文私钥（SK_{DP} 和 SK_{DU}）只能在 EKeyMgr 内解密。

一旦分析程序、加密数据和私钥准备就绪，数据分析任务就开始了。首先根据隐私描述语言（PDL）定义的规则检查分析程序。任何未通过此检查的程序都会导致分析任务立即终止。随后，EAnalyzer 加载并解密加密数据。然后验证解密数据的哈希值；如果与声称的哈希不匹配，EAnalyzer 立即停止以防止处理被篡改或欺诈的数据。如果数据得到验证，EAnalyzer 继续进行主要分析，最终使用 PK_{DU} 产生加密结果。此外，使用 SK_{DU} 对结果进行签名。由于 SK_{DU} 位于 EKeyMgr 内，EAnalyzer 通过本地认证建立安全通道来请求 SK_{DU}。

4.2 隐私描述语言

为了防止数据分析结果泄漏数据，我们提出了隐私描述语言来定义使用数据的规则，并通过了静态二进制分析的方法检查分析程序是否遵守隐私描述语言定义的规则。如图所示，这段代码使用 PDL 描述了在 Iris 数据集上使用 Kmeans 算法的规则，这段代码也将被转换成 LLVM 的中间表示，然后使用符号执行得到 PDL 描述的规则的状态 a。另一方面，我们使用 GTIRB 作为分析的中间表示，在将分析程序转换为中间表示后，使用符号执行获取每个输出变量的状态 b，并判断是否满足 b 包含于 a。

4.3 可信和可验证的结果

算法概述了 EKeyMgr 和 EAnalyzer 的主要操作。在 EAnalyzer 内生成的加密分析结果及其签名被传输到区块链。在那里，任何人都可以验证签名的有效性。有效的签名确认 DP 成功执行了分析程序并获得了正确的结果，因为签名源自 EAnalyzer 内部。这种机制防止攻击者在未执行分析程序的情况下伪造有效签名，从而确保数据分析过程的完整性。DU 可以从区块链下载加密结果，并使用 SK_{DU} 解密以获得明文分析结果。

4.4 一次性远程认证

我们实施一次性远程认证过程，从一开始就建立安全和可信的环境。在设置阶段，云服务提供商（CSP）通过与 Intel 认证服务进行远程认证来验证 EKeyMgr 生成的公钥 PK_{CSP} 的授权。这种彻底的验证确认了 CSP 凭据的完整性和合法性。一旦验证完成，CSP 上的后续分析任务不再需要远程认证。DP 和 DU 转发的私钥（SK）通过本地认证从 EKeyMgr 安全传输到 EAnalyzer。

5 评估

在本节中，我们通过综合实验评估 Fidelius 的性能。首先，我们通过分别执行 CPU 密集型和 I/O 密集型任务来评估其时间消耗。随后，我们对 Fidelius 与在以太坊虚拟机（EVM）中执行分析程序的替代解决方案进行性能比较。我们的实验在一台配备 Intel(R) Core(TM) i5-10400F CPU 的机器上进行，该 CPU 拥有 12 个核心，频率为 2.9 GHz，16 GB 内存和 12 MB 缓存。除非另有说明，每个实验重复 1,000 次以计算平均值。

5.1 CPU 密集型任务

为了评估 Fidelius 在 CPU 密集型任务上的性能，我们实现了一个具有三层和 128 个隐藏单元的神经网络算法，设计用于使用 MNIST 数据集识别手写数字。我们报告算法的时间消耗和准确性。此外，我们在原始 CPU（无 SGX）上部署算法进行比较。基于 Fidelius 的算法与基于原始 CPU 的算法之间的主要差异包括：

- 随机库。基于 Fidelius 的算法使用基于 Intel SGX 的随机库，而基于原始 CPU 的算法依赖 C++ 随机库。
- 数据解密。Fidelius 在启动算法前需要解密密封数据，产生额外的处理时间。相反，基于原始 CPU 的算法可以直接访问明文数据。

对于 Fidelius 和原始 CPU，我们保持固定的测试集数量为 1,000，同时将训练集数量从 10,000 逐步增加到 50,000。对于每个训练集，我们分别将 epoch 设置为 100 和 200 来执行算法。

图说明了在不同 epoch 下在 Fidelius 和原始 CPU 上运行的算法的时间消耗。随着训练集数量的增加，Fidelius 和原始 CPU 上算法的时间消耗都呈现线性增长趋势。然而，这两种算法之间的时间消耗差异很小。具体而言，观察到 Fidelius 上的执行时间仅比原始 CPU 长 2%，主要是由于数据解密开销。

图显示了在不同 epoch 下 Fidelius 和原始 CPU 上算法的准确性，两者都达到了超过 91% 的准确性。在 100 个 epoch 时，准确性略有波动，但平台之间没有明显差异。在 200 个 epoch 时，两种算法都收敛到几乎相同的准确性。Fidelius 与原始 CPU 相比时间消耗增加了 2%，但保持了相似的准确性水平，在 CPU 密集型任务中表现出可比的性能。

5.2 I/O 密集型任务

为了评估 Fidelius 在处理 I/O 密集型任务时的时间效率，我们在 Fidelius 和原始 CPU 平台上测试了一个在超过 1 GB 文件中搜索目标字符串的算法。时间消耗取决于数据加载和字符串搜索，Fidelius 由于数据解密需要额外时间。考虑到 Intel SGX 在 Fidelius 中的约束，我们将数据加载块大小设置为 64 KB 和 256 KB，并为每个块大小改变数据行数从 1 到 128 和 1 到 256 来测量时间效率。

图显示了 Fidelius 与原始 CPU 在不同数据块大小下的时间效率比较。随着读取行数的增加，Fidelius 的时间消耗显著减少。例如，在 64 KB 块大小时，Fidelius 的时间从 81 秒（1 行）下降到 11 秒（128 行）。类似地，在 256 KB 块大小时，从 213 秒（1 行）减少到 10 秒（256 行）。将块大小增加到 256 KB 进一步优化了 Fidelius 的性能。尽管有所改进，Fidelius 与原始 CPU 之间的时间消耗仍存在差距，主要是由于 I/O 密集型任务中的解密过程。

5.3 性能比较

在本节中，我们将 Fidelius 与 SDTE 的性能进行比较，SDTE 使用 k-最近邻（k-NN）算法作为以太坊智能合约在以太坊虚拟机（EVM）上执行机器学习任务。我们在 Fidelius 和原始 CPU 上实现 k-NN 算法来评估每个解决方案的时间消耗，使用来自 Kaggle 的 Titanic 数据集作为输入。

为了评估 k-NN 算法在以太坊虚拟机（EVM）上的执行时间，我们在 Ganache（个人以太坊区块链）上部署了 k-NN 智能合约。报告的 EVM 上 k-NN 的时间消耗仅关注执行时间，不包括交易广播或提交所花费的时间。我们还排除了将 Titanic 数据集上传到区块链所需的时间，

因为 k-NN 合约使用链上数据。由于 EVM 的内存限制，k-NN 算法无法处理超过 60 个测试集，因此我们将测试集限制在 10-60 个，同时保持训练集为 100 个。

图显示了 k-NN 算法在原始 CPU、Fidelius 和以太坊虚拟机（EVM）上的时间消耗，后者比其他的大约长 30 倍。此外，由于内存约束，EVM 执行在测试超过 60 个集时经常失败，突出了在 EVM 上运行具有大数据集的复杂机器学习算法的困难，正如之前关于以太坊内存限制的研究所指出的那样。相比之下，Fidelius 为复杂的机器学习任务提供了更可靠和稳定的环境。

6 相关工作

GXS 作为一个基于区块链的数据交易平台运行，记录并促进买卖双方之间的交易，双方在相互同意后通过私有渠道交换数据。AccountTrade 使用区块链将分析器与数据提供方配对，并通过可执行的协议和数据索引确保生态系统安全，这些协议和索引检测并惩罚不诚实的行为。Zhao 等人开发了一个专注于提供方隐私的数据交易系统，采用环签名和双重认证来保护交易并防止未授权访问。

在数据共享系统中，Shen 等人开发了一个确保数据完整性和机密性的方案。Zuo 等人引入了一个系统，使用代理重加密和密钥分离高效保护和撤销卖方的密钥，通过基于属性的加密增强数据保护。为了缓解恶意代理参与数据泄露，Guo 等人提出了可问责代理重加密（APRE），这是一个检测和解决重加密密钥滥用的框架，进一步验证了其在 DBDH 假设下的 CPA 安全性和可问责性。Deng 等人形式化了一个基于身份的加密转换（IBET）模型，用于与数据所有者初始指定之外的额外接收者共享加密数据。

SDTE 引入了一个基于区块链的数据分析平台，数据提供方加密数据并上传到区块链。数据使用方选择这些数据，形成分析合约并请求服务。获得批准后，提供方通过 Intel SGX 远程认证将解密密钥发送给可信节点，然后该节点解密数据并在 Intel SGX 保护的以太坊虚拟机（EVM）中运行分析，然后将结果上传到结算合约。然而，SDTE 的加密数据上传对区块链施加了显著的存储需求，并且作为以太坊智能合约运行分析引入了性能挑战。在 EVM 上运行复杂的算法如 k-NN，由于数据规模和任务复杂性，通常是不实际的。

PrivacyGuard 通过将链上分析转移到链外可信执行环境（TEE）来缓解以太坊虚拟机（EVM）的性能限制，允许数据提供方设置限制未授权数据使用的策略。然而，它仍然遇到 EVM 性能的挑战。类似地，Sterling 引入了一个利用机器学习进行数据分析的去中心化数据市场，但也像 SDTE 和 PrivacyGuard 一样对区块链施加了显著的存储需求。此外，隐私担忧或法规可能阻止敏感数据的上传。

7 结论

Fidelius 是一个新颖的系统，在复杂的多角色场景中增强数据分析安全性。利用 Intel SGX 和区块链技术，它解决了数据泄露问题并保证可信、可验证的计算结果。通过采用静态二进制分析和隐私描述语言（PDL），Fidelius 保护结果，而其密码协议确保完整性。本地认证一致地验证分析程序，减少对中心化服务的依赖并提高效率。实验表明，Fidelius 产生的开销最小，同时超越现有解决方案，为在不同环境中保护敏感数据提供了强大的方法。