外包多所有者秘密共享数据库上的语义安全的私有集合求交

Dongfang Zhao内
华达大学雷诺分校，美国
dzhao@unr.edu

arXiv:2303.06863v1 [cs.CR] 13 Mar 2023

摘要

隐私集合求交（PSI）旨在让用户找出用户之间共同拥有的项目，同时不泄露其他成员信息。数据库领域最新提出的 PSI 方法是 Prism，该方法基于秘密共享，并假设有多个不串通服务器可用。Prism 的一个局限在于其语义安全性：服务器上的编码是确定性的，这意味着该方案在选择明文攻击下不可区分（IND-CPA）。本文通过两个正交原语扩展了 Prism 原有的隐私集合求交方案，即 Kaleido-RND 和 Kaleido-AES：前者采用随机化编码，性能极高；后者在选择明文攻击下具有可证明的安全性，但计算开销更大。本文实现并部署了一个系统原型，该原型运行在由 34 个 SQLite 实例组成的集群上。针对 TPC-H 基准测试和三个实际应用开展的大量实验，验证了所提出的 Kaleido 原语的有效性。

ACM 引用格式：
Dongfang Zhao。2023年。外包多所有者秘密共享数据库上的语义安全的私有集合求交。载于*ACM* 会议论文集（*Conference'23*）。ACM，纽约，纽约州，美国，14 页。
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 引言

隐私集合求交（PSI）旨在让用户找出彼此共有的项目，同时不泄露其他成员信息。作为一种用途广泛的密码学原语，隐私集合求交已在应用密码学文献中得到广泛研究 [39] ，并且近期也出现在隐私保护外包数据库领域 [30]。具体而言，隐私集合求交已被视为支持外包数据库中各种操作的重要构建模块，例如半连接 [24]。

数据库领域最近提出的隐私集合求交方法是 Prism [30], ，它基于秘密共享，并假设存在多台互不串通的服务器。顾名思义，秘密共享 [46], 允许用户将明文拆分为多个部分，使得单个部分（存储在不同的服务器上）不会泄露有关明文的任何信息。秘密共享通常被视为安全多方计算（MPC） [49]的一种特定形式。Prism 的高效率基于群同态指数运算。

Prism [30]的一个局限在于其语义安全性：尽管通过数论模幂掩盖了共享实体的确切数量，但这种编码是确定性的，这意味着该方案无法达到选择明文攻击下的不可区分性（IND-CPA）。因此，本文的首要目标是描述一种攻击，该攻击能够揭示参与方交集之外的信息。

为保证 Prism 的语义安全性，本文的第二个目标是通过两种相互正交的原语扩展 Prism 原有的隐私集合求交方案：(i) *Kaleido-RND*，一种高效模块，用于随机化群生成元，并以此在乘法群的陪域中进行幂运算；以及 (ii) *Kaleido-AES*，一种伪随机函数，用于确保整个隐私集合求交方案具备 IND-CPA安全性。正如其名称所示，我们提出的方案旨在对原有的 Prism 方案进行混淆，最终形成一个散射状的棱镜，看起来如同万花筒。虽然 Kaleido-RND 效率很高，且确实能对密文进行随机化，但尚未证明 Kaleido-RND 具有 IND-CPA安全性。不过，我们将证明 Kaleido-AES 在付出计算开销的代价下，能够实现可证明的语义安全性。

为展示所提出的 Kaleido 方案的有效性，我们使用约 1,000 行 Python 和 Shell 脚本实现了一个系统原型，并在 CloudLab [9] 上由 34 个 SQLite 实例组成的集群中部署该系统。我们使用 TPC-H 基准测试和三个实际应用评估 Prism、Kaleido-RND 和 Kaleido-AES，其中我们

允许免费制作本作品全部或部分内容的数字或纸质副本，供个人或课堂使用，前提是这些副本不得用于营利或商业利益，且每份副本均须附上本声明和首页的完整引文。对于本作品中归他人而非 ACM 所有的组件，必须尊重其版权。注明出处后，允许摘录。若要以其他方式复制、重新发表、发布到服务器或分发至列表，须事先获得明确许可和/或支付费用。请向 permissions@acm.org 申请许可。
Conference'23，*2023*年*6*月，美国华盛顿州西雅图© 2023年 计算机协会。ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

将数据库所有者（即客户端）的数量设为 2 到 32。结果表明，Kaleido-RND 仅给服务器编码带来微不足道的开销，在整个系统中的开销通常可以忽略不计。此外，尽管与原始 Prism 相比，Kaleido-AES 似乎引入了较大的计算开销，但其开销比通信成本低几个数量级，这证明了 Kaleido-AES 除具备可证明安全性外，也具有实用性。

总之，本文作出了以下技术贡献。

- 我们揭示了外包数据库中一种最先进的隐私集合求交方案 Prism 存在安全漏洞。我们阐述了有效攻击背后的直觉，并通过具体示例详细说明了这种攻击。（§3）
- 我们提出了一系列新原语，称为 Kaleido，用以扩展 Prism，使新的隐私集合求交方案具备语义安全性。我们证明了 Kaleido 在广泛认可的计算假设下具有选择明文攻击下的不可区分性，并通过理论分析和案例研究验证了其正确性。（§4）
- 我们使用约 1,000 行 Python 和 Shell 脚本实现了 Kaleido 方案。我们将系统部署在公有云 CloudLab [9] 上，其中包含两台服务器和 32 个数据库所有者。针对 TPC-H基准测试和三个真实应用开展的大量实验，证实了所提出的 Kaleido 方案的有效性。（§5）

2 预备知识

2.1 隐私集合求交与安全多方计算

隐私集合求交（PSI） [11, 17, 23, 27, 39, 40, 42] 已在外包数据库中得到广泛研究，例如作为构建模块，用于外包数据库之间的隐私保护连接 [2, 33]。事实上，PSI 可视为多方之间集合运算的一种特殊形式，因此可以通过通用安全多方计算方案实现；这类方案主要依赖算术门或布尔门，会带来显著的计算开销。

安全多方计算（MPC） [1, 13, 32] 有着悠久的历史 [49]。MPC 的目标更为宏大：除了确保明文机密，我们还希望仅通过接触多个参与方持有的编码数据，就能计算原始明文的任意函数。最初的问题是通过所谓的混淆电路 [49], 解决的，其思路相当简单：我们可以让每个参与方用自己的私钥对输入进行编码，打乱加密后的密文，然后枚举所有密钥来解密结果。由于我们假设加密方案是安全的，因此只有将正确的私钥组合应用于某个混淆输出，才能揭示结果。这确实是一种可行的解决方案，至少在理论上如此；但在实践中，电路可能呈指数级增长，从而导致效率问题。还有许多效率更高的 MPC 解决方案，例如 [3, 26, 34, 50]。

PSI方案的主要局限之一在于其可扩展性。事实上，许多现有的PSI方案仅支持两方，例如 [6, 20, 41]。较新的研究 [7, 15, 19,25, 28–30] 则致力于开发能够高效支持两方以上的PSI方案。本研究属于后者：所提出的Kaleido方案支持任意数量的参与方执行PSI操作。

2.2 秘密共享

秘密共享方案（SSS）的原理很简单：给定的明文 pt 会被转换为一组编码字节ct，只有特定的子集 ct才能重构原始 pt。SSS 的目标是降低明文泄露的风险；恶意对手无需攻破明文持有者，而是必须在任何份额持有者发现攻击之前攻破多个实体。即使在假设对手较弱、仅为半诚实对手的情况下，将秘密份额分散给更多参与方，也能提高成功实施窃听攻击的门槛。

在实践中，可以通过子集大小调整 SSS。形式上，$(t,\ n)$-门限秘密共享方案（TSSS）定义如下。

定义 1（TSSS）。A（$t,\ n$）-*TSSS* 由两个算法组成：

- 份额生成：一种随机化算法，以明文 pt 为输入，并返回一列份额 $S = (s_1, \ldots, s_n)$ 。
- 重构：一种确定性算法，输入至少 t 个份额并返回明文。

数值 t 称为 *TSSS* 的门限。令 U 为 n 个份额中大小为 t 的子集，$|U| \geq t$ 且 $U \subseteq \{s_1, \ldots, s_n\}$，我们要求 *TSSS* 满足以下性质：

$$Reconstruct(U) = pt.$$

正如我们将在下一节 §2.3 中看到的，TSSS 的定义会导致其安全定义与传统加密方案的安全定义略有不同。

(t, n)-TSSS 的典型示例来自 Shamir [47],，其中秘密通过一个 $(t-1)$ 次多项式揭示。本质上，每个份额都可以通过拉格朗日多项式重构未知数中特定次数项的系数。除了 Shamir 的构造外，还存在其他方案。Ito等人 [21] 提出了复制秘密共享方案，该方案基于有限域，其中每个份额都是一个向量。复制秘密共享的一个优点是其线性性：局部

份额的加法和减法可以线性转换为明文的加法和减法。复制秘密共享的一个更简单的变体是加法秘密共享，其中每个份额都是一个标量值，门限 t 设为 n。

2.3 可能安全性

在应用中使用加密方案时，最好能够以可证明的方式证明其安全性。形式上，要证明给定加密方案的安全性，我们需要确定以下三个重要要素：安全目标、威胁模型和假设。安全目标阐明应用遭受攻击时所期望达到的效果；威胁模型描述对手能够如何发起攻击，例如可以收集哪些明文或密文信息，以及攻击所受的资源和时间限制；假设则列出密码方案所依赖的预设条件（例如，对两个大素数的乘积进行因数分解，求解离散对数）。安全目标和威胁模型通常合称为安全定义。

一种广受认可、兼顾效率与安全性的安全定义是：对手可以发起选择明文攻击（CPA），定义如下。

定义2（选择明文攻击）。给定安全参数 n，即密钥的比特串长度，对手最多可以获得 $poly(n)$ 组明文-密文对 (m, c)，其中 m 由对手任意选择，而 $poly(\cdot)$ 是关于 n 的多项式函数。利用这些信息，对手试图解密一个 c'，而它不属于已知密文的多项式数量范围。

多项式要求规定，对手只能运行多项式时间算法，而不能拥有无限资源。因此，我们希望设计具有*CPA*安全性的加密方案：即使对手 A 能够获得这些额外信息，A 解码密文的能力也不应超过随机猜测，除非概率差异极小。为了量化这种小概率的程度，下面定义可忽略函数。

定义 3。如果对于所有多项式 $poly(n)$，不等式 $\mu(n) < \frac{1}{poly(n)}$ 在足够大的 n 值下成立，则称函数 $\mu(\cdot)$ 为可忽略函数。

为完整起见，我们列出以下关于可忽略函数的引理，后续章节将会用到。这里不提供证明，相关内容可参见密码学或复杂性理论的入门教材。

引理*1*（两个可忽略函数之间进行算术运算，结果仍为可忽略函数）。设 $\mu_1(n)$ 和 $\mu_2(n)$ 均为可忽略函数，则以下函数均为可忽略函数：$\mu_1(n) + \mu_2(n)$、$\mu_1(n) - \mu_2(n)$、$\mu_1(n) \times \mu_2(n)$，以及 $\mu_1(n)\,\mu_2(n)$。

证明所提出的加密方案（例如选择明文攻击下的不可区分性）的安全性的标准方法是通过归约 [31]。通常，将破解该方案归约为解决一个困难的数学问题，这意味着如果能够攻击该方案，就可以高效地解决该数学问题。也就是说，该加密方案至少与该数学问题一样难。该方案被建模为一个子程序，其输入经过模拟，使对手无法分辨自己是在参与攻击，还是在协助解决困难问题的子程序中。尽管可以进行正向证明，但更常用的技术是反证法：假设对手能够在某个实验中以非可忽略的优势区分两个密文，那么归约就能以非可忽略的概率高效地解决一个被认为难以求解的数学问题，从而导致矛盾。

3 PRISM PSI

本节讨论 Prism [30] 可能存在的安全问题。我们首先介绍 Prism 隐私集合求交方案（§3.1），然后进行安全性分析（§3.2），最后通过一个现实案例说明这些安全问题（§3.3）。

3.1 Prism 隐私集合求交方案的定义

本小节简要介绍 Prism [30]。

3.1.1 假设。

不串通服务器。 Prism 假设存在一个由不合谋服务器组成的集群，用于存储由原始明文生成的秘密份额。在实践中，可以通过从不同的云计算服务商（例如 Google Cloud Platform 和 Amazon Web Services）获取虚拟机来实现这一假设。

离散对数问题的困难性。 设 $\mathbb{G}$ 表示一个（乘法）循环群，g 是其生成元。也就是说，$\forall y \in \mathbb{G}$，$\exists x \in [0, |\mathbb{G}| - 1]$，使得 $y = g^x$。直观而言，离散对数（DL）问题指出，如果 $\mathbb{G}$ 的阶足够大，那么即使给定 y 和 g，也无法在多项式时间内找到 x。这是大多数现代密码方案（如 ECDSA [22] 和 ElGamal [10]）普遍接受的假设。

客户端知识。 我们假设客户端了解执行隐私集合求交所依据的属性域。在隐私集合求交的场景中，我们主要关注类别数据或离散数据 [30],，这意味着每个客户端都可以用布尔向量表示其本地项目，其中每个元素都是一个布尔值，用于指示客户端上是否存在某个特定项目。

3.1.2 架构。 完整部署的 Prism 包含四个子系统：客户端（即数据库所有者）、不合谋的服务器、预言机（发起方）和公告方。不过，隐私集合求交功能可通过前三个子系统完成，无需公告方参与。

客户端（数据库所有者）。客户端是敏感数据的所有者，这些数据应对其他客户端保密。我们假设客户端在隐私集合求交过程中是诚实的：（i）客户端没有在隐私集合求交中谎报本地数据项的动机，也无意探查或分析任何中间结果；（ii）客户端不会在未被发现的情况下遭到对手入侵，这可以通过数字签名 [22] 和拜占庭容错协议 [5]来实现。

服务器。在 Prism 的隐私集合求交方案中，有两台服务器，且它们不会串通。这意味着服务器不可能是主动对手；也就是说，两台服务器都不是恶意的。然而，服务器并非完全可信，因为它们可能作为被动对手，试图通过参与外包数据库来了解明文。由于两台服务器各自接触到的数据都是由秘密份额生成协议随机产生的，每台服务器都无法探查明文，因此该方案在信息论意义上是安全的。

预言机。 预言机也称为发起方，负责为客户端和服务器设置初始参数。例如，Prism 需要两个群，而群阶和生成元由预言机指定。预言机仅在隐私集合求交方案开始时发挥作用；如果上下文明确，我们会省略预言机的初始化步骤，并假定该步骤已在离线阶段完成。需要指出的是，Prism 中的预言机 并非密码学中广泛使用的传统预言机：在后一种情况下，预言机通常指功能“理想化”且不透露任何内部机制的实体，也就是说，它是一个黑盒。

图 1 展示了多所有者外包数据库上的隐私集合求交系统模型。我们假设有四个客户端 $\{C_0,\ C_1,\ C_2,\ C_3\}$ 和两台不串通服务器 $\{S_0,\ S_1\}$。在客户端开始外包本地数据之前，预言机会向客户端和服务器发送参数。随后，客户端对本地数据应用加法秘密共享基元，并通过安全通信信道（例如 OpenSSL [36]）将其发送给两台服务器。两台服务器执行协议，对秘密份额进行编码，并将编码值广播回所有客户端。

3.1.3 协议。

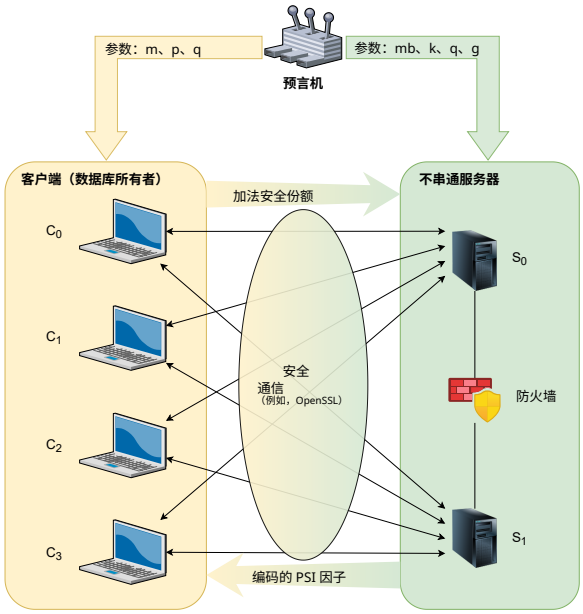


图 1：多所有者数据库上的隐私集合求交系统模型

初始化。预言机选取两个循环群的参数，并将其广播给服务器和/或客户端。其中一些参数（例如乘法群的生成元）应仅对服务器保密，以确保客户端无法获知隐私集合求交之外的任何信息。除了这些参数外，预言机还负责配置平台，例如主机名和其他元数据信息。

属性域的向量化。 每个客户端 C_i 都会生成一个本地向量 V_i，使得 $\forall v \in V_i \implies v \in \{0,1\}$。当且仅当 C_i上存在对应项时，V_i中的一个值$v = 1$；否则 $v = 0$。事实上，我们在此假设所有客户端对属性域中各项采用相同的顺序，记为 A_c。一种直接确定域值顺序的方法是采用字母数字顺序。

秘密共享。 每个客户端 C_i 都会将其本地向量 V_i 拆分为两个秘密向量，分别记为 V_i^0和 V_i^1，并分别发送给 S_0 和 S_1。拆分的一个要求是，秘密必须具有可加性，这意味着对于 $V_i^b, b \in \{0,1\}$，必须能够通过对秘密执行加法运算来重构 V_i 。Prism 利用的可加性是加法群 $\mathbb{Z}_p = \{0,1,\ldots,p-1\}$ 中定义的模加法，其中 p表示一个大素数。也就是说，给定明文 $v \in \{0,1\}$[1]，，我们随机选取第一个份额

[1]请注意，向量的元素都是布尔值，即 0 或 1。

$v^0 \in [0, m)$，并按如下方式计算第二个份额 v^1：

$$v^1 \equiv v - v^0 \pmod{\mathrm{p}}.$$

如果我们将上述拆分应用于本地向量 V_i 的所有元素，就可以将上述标量等式推广到向量，并按元素进行运算，如下所示：

$$V_i^1 \equiv V_i - V_i^0 \pmod{\mathrm{p}}.$$

聚合份额。服务器 S_b 收到所有秘密份额 $V_i^b, b \in \{0, 1\}$ 和 $0 \le i < m$, S_b 后，会对所有客户端 C_i 的秘密份额进行本地聚合。也就是说，服务器 S_b 按如下方式计算一个新向量 V^b：

$$V^b \equiv -[m^b] + \sum_{i=0}^{m-1} V_i^b \pmod{\mathrm{p}}, \tag{1}$$

其中，$[m^b]$ 表示根据客户端数量 $m, b \in \{0, 1\}$生成的秘密份额的扩展向量（长度与 V^b 相同），其计算方式与我们先前计算 v^b的方式相同。添加 $-m^b$ 的操作数，是因为我们希望在后续步骤中消去表示共享属性值存在与否的向量中的值"1"。

编码份额。服务器不应直接将本地聚合后的份额发送回客户端，因为这样会泄露客户端之间共享项目的确切数量。相反，服务器必须以某种方式对部分份额进行编码，使客户端除了交集之外无法获知共享项目的信息。毫不意外，Prism采用的编码方案利用了两个循环群$\mathbb{G} = (G, +)$和 $\mathbb{H} = (H, \times)$之间同态函数 Hom 的代数性质，使得：

$$\begin{aligned} Hom : G &\to H \\ x &\mapsto g^x \pmod{\mathrm{q}}, \end{aligned} \tag{2}$$

其中，g是生成元，q是乘法群 $\mathbb{H}$的阶。显然，函数 $Hom()$是同态的，$\forall x_0, x_1 \in G$：

$$Hom(x_0) \times Hom(x_1) = g^{x_0} \times g^{x_1} = g^{x_0 + x_1} = Hom(x_0 + x_1).$$

然后，我们可以在服务器S_b上对 V^b的元素应用 $Hom()$。令 U^b 表示编码向量，其中每个元素都应用了 $Hom()$函数。服务器 S_b向所有客户端广播 U^b。

计算隐私集合求交。每个客户端分别从两台服务器 S_0 和 S_1 接收两个向量 U^0 和 U^1。客户端在不知道 g（乘法群的生成元）值的情况下，计算 U^0 和 U^1 的逐元素乘积。设 n 表示 A_c 的定义域基数；若 $U^0 = (u_0^0, u_1^0, \ldots\ldots, u_{n-1}^0)$ 且 $U^1 = (u_0^1, u_1^1, \ldots\ldots, u_{n-1}^1)$，则我们将逐元素乘积定义如下：

$$U = U^0 \odot U^1 \stackrel{\text{def}}{=} \left(u_0^0 \cdot u_0^1, u_1^0 \cdot u_1^1, \ldots, u_{n-1}^0 \cdot u_{n-1}^1\right). \tag{3}$$

显然，$\forall i \in [0, n)$，当且仅当所有客户端的秘密份额之和等于 m 时，$u_i \text{def}= u_i^0 \cdot u_i^1 = 1$。实际上，这是检测属性值交集的判据：只有所有 m 个客户端的编码均为 1 的值，才应纳入隐私集合求交。另一方面，如果布尔密文不全为 1，那么 u_i 就不能为 1，这意味着属性域中的第 i 个值不属于隐私集合求交。此外，由于只有服务器能看到 g，客户端无法获知共享属性域中第 i 个值的客户端确切数量。

3.2 Prism的安全性分析

3.2.1 高层直觉。除非每个客户端都持有属性域的第 i个值，否则Prism会隐藏共享该值的客户端的确切数量。本节将说明，Prism方案还会泄露其他信息。关键在于，式（2）中的函数是确定性的，这意味着相同的输入会产生相同的输出。因此，如果共享某个特定属性值的客户端数量相同，Prism就会向对手泄露这一信息。

3.2.2 选择明文攻击。

安全性概念。通俗来说，选择明文攻击（CPA）是指这样一种攻击：即使在安全参数（通常为问题规模的比特串长度，例如属性域的基数）的多项式次运算时间内运行攻击算法后，对手仍能以不可忽略的概率区分任意选择的两个明文所对应的密文。以 n为参数的可忽略函数定义为

$$negl(n) \stackrel{\text{def}}{=} o(n^{-c}), \tag{4}$$

其中 c 表示任意常数。这等价于：

$$\forall c > 0 \implies \lim_{n \to \infty} \frac{negl(n)}{n^{-c}} = 0.$$

方案 Π 的*CPA*下的不可区分性（IND-CPA）形式化证明通常采用模拟的形式：假设 Π 不具备 IND-CPA 安全性，并利用 Π 构造一个能够解决难解问题的协议，从而导出矛盾。在 PSI 的语境下，IND-CPA 意味着，U 中除一个表示共享属性值的元素外，所有元素看起来都应当是随机的。对手只需两轮就能轻易破坏 Prism 的 IND-CPA 安全性：

- 对手向除第一个客户端外的所有客户端插入一个新的属性值 x_n，并从任意客户端检索出 U中对应的元素，例如 u_n。
- 对手向除第一个客户端之外的所有客户端插入一个新的属性值 x_{n+1}，并向除前两个客户端之外的所有客户端插入另一个新的属性值x_{n+2}。与第一轮类似，对手在 U 中获取两个密文：u_{n+1} 和 u_{n+2}。

对手通过比较 u_n 与 u_{n+1} 和 u_{n+2}，即可必胜选择明文攻击下的不可区分性游戏。尽管有人可能会质疑允许对手向数据库所有者插入新的属性值是否切实可行，但 CPA 安全性确实考虑了"最坏情况"。即使这种选择明文攻击下的不可区分性可能只具有理论上的意义，下一节仍将构造一种更实际的攻击。

3.2.3 超越*PSI*的推断。假设每个客户端根据 $C_i, i \in [0, m)$ 计算编码向量 $U = (u_0, u_1, \ldots\ldots, u_{n-1})$，其中 n 表示属性域的基数。不失一般性，设 $0 \le i \ne j < n$ 和 $u_i = u_j \ne 1$。这意味着属性 A_c 的第 i 个值和第 j 个值最多分别由 $m-1$ 个客户端持有。设 $[\cdot]$ 表示数组中的位置元素，特定客户端 C_k 可以推断出以下信息：

- 如果 $V_k[i] = V_k[j]$，那么 C_k 知道 $A_c[i]$ 和 $A_c[j]$ 在所有数据库所有者中的"受欢迎程度"相同。
- 如果 $V_k[i] < V_k[j]$，即 $V_k[i] = 0$ 和 $V_k[j] = 1$，那么 C_k 知道恰好还有一个客户端持有 $A_c[i]$ 而非 $A_c[j]$。根据具体应用，此类信息可能会引入偏见或不公平，例如在竞争者市场中。
- 如果 $V_k[i] > V_k[j]$，客户端 C_k 可以推断出与上述第二种情况对称的类似信息。

显然，上述三种情形中的任何一种都会泄露超出客户端之间项目交集的信息。

3.3 Prism 上的 CPA 攻击

本节通过一个运行示例展示 Prism 的 PSI协议及针对它的攻击。

3.3.1 系统配置。 我们假设 $m = 4$，即有四个客户端，C_i、$0 \le i < 4$。除非另有说明，令 $b \leftarrow \{0, 1\}$，即 b 以 50% 的概率取 0 或 1[2]。不串通服务器的数量固定；为简单起见，设有两台服务器：S_0 和 S_1。令加法群 $\mathbb{G} = (\mathbb{Z}_5, +)$、乘法群 $\mathbb{H} = (\mathbb{Z}_{11}^*, \times)$，并令 $g = 3$ 为 $\mathbb{H}$的一个子群的生成元。请注意，上述参数仅为便于说明而选取；在实际应用中，这些参数通常要大得多，以抵御暴力破解攻击。

3.3.2 模式和数据。

模式与元数据。 为简单起见，我们假设四个客户端各自持有的本地关系均采用单属性模式。由于只有一个属性，我们可以放心地将该关系的列表示为向量。我们还假设该单一属性的数据类型为半开区间 $[0, 5)$ 内的整数。我们用 n 表示属性域的基数，即在我们的示例中为 $n = 5$。这意味着，尽管每个客户端可以存储任意数量的值，但每个明文向量的长度都是 5，即 $|V_i| = 5, 0 \le i < 4$。

数据。 令 R_i 表示客户端C_i 上的单属性关系，形式为列表：

$$\begin{cases} R_0 = (0, 1, 3) \\ R_1 = (1, 3, 4) \\ R_2 = (3, 4, 4) \\ R_3 = (1, 2, 3, 4) \end{cases} \tag{5}$$

隐私集合求交的目标是让每个 C_i 找出共同拥有的元素，在本例中为整数 3。

3.3.3 向量化。 式 (5) 中四个客户端上的明文数据可以向量化为：

$$\begin{cases} V_0 = (1, 1, 0, 1, 0) \\ V_1 = (0, 1, 0, 1, 1) \\ V_2 = (0, 0, 0, 1, 1) \\ V_3 = (0, 1, 1, 1, 1) \end{cases} \tag{6}$$

应纳入隐私集合求交的元素，是那些位置向量元素为 1 的值。在本例中，唯一符合条件的元素是整数 3。若将隐私集合求交表示为定义在关系集合上的函数，则有

$$PSI(\mathcal{R}) = \{3\},$$

其中 $\mathrm{R}\ \mathrm{def} = \{R_i\}$，$0 \le i < m$。我们还引入另一个参数化函数 $Card(\cdot\ k)$，它返回由 k 客户端共享的属性值集合：

$$\begin{aligned} Card : R^m \times \mathbb{Z}_m &\to 2^{A_c} \\ (\mathcal{R}, k) &\mapsto \{y \mid y \subseteq A_c\}, \end{aligned}$$

其中，R 表示一个介于 0 和 n 之间的任意整数向量，R^m 表示在 R上进行 m 次笛卡尔积。为简便起见，我们有时将第二个参数置于下标中，即 $Card_k(\cdot)\ \mathrm{def} = Card(\cdot\ k)$。根据定义，显然 $PSI(\mathrm{R})$是 $Card_k(\mathrm{R})$的特例，即 $PSI(\mathrm{R}) \equiv Card_{|\mathcal{R}|}(\mathrm{R})$。采用这一记法，我们可以将示例写为：

$$\begin{cases} Card_0(\mathcal{R}) = \emptyset \\ Card_1(\mathcal{R}) = \{0, 2\} \\ Card_2(\mathcal{R}) = \emptyset \\ Card_3(\mathcal{R}) = \{1, 4\} \\ Card_4(\mathcal{R}) = \{3\} \end{cases} \tag{7}$$

[2]在密码学文献中，也常用 $\$\leftarrow$、$\leftarrow_R$ 或其他形式来表示同一事物。

我们注意到，某些 $Card_k()$的基数相同，例如 $|Card_1(R)| = |Card_3(R)| = 2$。正如我们稍后将看到的，这会泄露可被对手利用的信息。

3.3.4 秘密共享。回想一下，客户端依赖加法群 $\mathbb{G} = (\mathbb{Z}_5, +)$ 进行加法秘密共享，如 §3.1.3 所述。因此，式（6）中的 V 可以拆分为如下形式：

$$\begin{cases} V_0^0 = (3,4,1,2,0);\ V_0^1 = (3,2,4,4,0) \\ V_1^0 = (1,2,4,3,4);\ V_1^1 = (4,4,1,3,2) \\ V_2^0 = (0,4,2,3,1);\ V_2^1 = (0,1,3,3,0) \\ V_3^0 = (2,3,4,1,0);\ V_3^1 = (3,3,2,0,1) \end{cases}$$

可以验证，在 $\mathbb{G}$ 中，$V_i = V_i^0 + V_i^1 = 0 \le i < 4$。客户端 C_i 随后将 V_i^0 发送给服务器 S_0，并将 V_i^1 发送给服务器 S_1。

3.3.5 服务器编码。两台服务器各自接收 m =4 个份额向量。也就是说，服务器 S_b 接收 V_i^b、$b \in \{0, 1\}$、$0 \le i < m$。如式（1）所述，m 也应在 $\mathbb{G}$ 上拆分。在此示例中，令 $m^0 = 1$ 和 $m^1 = 3$。因此，服务器 S_0 上聚合向量 V^0 的第一个元素可计算为

$$V^0[0] = -m^0 + \sum_{i=0}^{3} V_i^0 = -1 + (3+1+0+2) \equiv 0 \pmod 5.$$

同样，我们可以计算两个聚合向量

$$\begin{cases} V^0 = (0,2,0,3,4) \\ V^1 = (2,2,2,2,0) \end{cases}$$

随后，两个服务器将式（2）应用于 V^b，并生成两个新向量（注意，在我们的示例中，$q = 11$）：

$$\begin{cases} U^0 = (1,9,1,5,4) \\ U^1 = (9,9,9,9,1) \end{cases}$$

随后，两个服务器将 U^b 广播给所有客户端。

3.3.6 客户端解密。所有客户端都从两个服务器接收两个向量 U^b。每个客户端在群 $\mathbb{H}$（即模 11）下计算 U^b 的逐元素乘法。例如，

$$U^0[1] \times U^1[1] = 9 \times 9 = 81 \equiv 4 \pmod{11},$$

and

$$U^0[3] \times U^1[3] = 5 \times 9 = 45 \equiv 1 \pmod{11}.$$

因此，每个客户端上的最终向量为

$$U = (9,4,9,1,4).$$

此时，所有客户端都知道第四个值 A_c 由所有客户端共享，因为 $U[3] = 1$；但由于根据式（2），这些加密值毫无意义，他们完全不知道其他客户端的数量，也不知道其他客户端持有哪些值。

3.3.7 信息泄露。所有客户端都得知 $U[0] = U[2] = 9$。让我们看看客户端 C_0 能从中了解到什么。回想一下，客户端 C_0 知道自己的本地向量，例如 $V_0[0] = 1$ 和$V_0[2] = 0$。这意味着 C_0 可以推断出以下事实：至少有一个客户端 $C_j, j \neq 0$，使得 $V_j[2] =1$；否则 $U[0]$ 就不可能等于 $U[2]$。也就是说，通过遵循 Prism PSI 方案，C_0 得知整数“2”存在于至少一个其他三个客户端中，尽管“2”并非由全部四个客户端共享（因为 $U[2] = 9 \neq 1$）。

4 KALEIDO PSI

为解决 Prism PSI 的信息泄露问题，我们针对多所有者数据库设计了一种新的隐私集合求交方案，称为 Kaleido。顾名思义，Kaleido 会在多所有者数据库之间的隐私集合求交计算中引入混乱，因此可视为“经过混淆的”Prism。Kaleido 的核心思想相当简单：不使用固定生成元，而是将其随机化，使得服务器上的编码份额即使对应相同的输入，也无法被区分。应用这一思想所面临的技术挑战，是确保隐私集合求交的正确性和安全性。

4.1 Kaleido服务器协议

说明。 我们详细说明算法1中 Kaleido服务器引入的随机性。第 1–7 行只是汇总所有客户端发送的秘密份额，并减去表示客户端数量的秘密份额 m。第 10 行确保：(i) 参数化函数的输出与随机字符串不可区分；(ii) 两台服务器上的输出相同，因为两台服务器共同商定了密钥 k。在实践中，$PRF_k(\cdot)$ 可通过分组密码实现，例如 AES [35]。随后，Kaleido 将随机化后的输出映射到群 $\mathbb{H}$中的一个元素，这一步在第 11 行完成。与第 10 行一样，S_0 和 S_1 都会生成相同的 g，因为 $\mathbb{H}$ 已公开给两台服务器。第 12–14 行确保 g 是阶为 p的（子）群的生成元。请注意，我们用 $|g|$（第 12 行）表示 g 在循环群中的阶，而不是标量 g 在算术运算中的绝对值。在第 15 行，群 $\mathbb{H}$ 中的幂运算结果取代了群 $\mathbb{G}$中的元素。第 16 行确保，即使 $\mathbb{G}$ 中的元素相同，由于 ct 和 g的取值不同，它们映射到 $\mathbb{H}$ 中的元素也各不相同。最后，服务器 S_b 将其本地编码 V^b 广播给所有客户端，如第 18 行所示。

图2展示了一个简单示例：两台服务器都采用初始向量（初始化向量）对客户端提交的秘密份额进行随机编码。直观而言，由于两台服务器使用执行开始时由预言机初始化的同一密钥 k 来配置伪随机函数 $PRF(\cdot)$，因此基本数值 g（参见算法1第11行）在两台服务器上相同，但在不同轮次中看起来是随机的。

算法1： *Enc( )*：Kaleido PSI 服务器协议

输入： 两台服务器 S_0 和 S_1； m 个客户端 C_i，$0 \le i < m$，每个客户端向 $S_b, b \in \{0,1\}$ 发送 V_i^b；向量 V_i^b 的长度为 n；预言机 O 将 m 拆分为 m^b，使得 $m \equiv m^0 + m^1 \pmod p$，并且将 m^b 发送给 S_b；共享密钥 k 由双方协商依赖于 S_b；一种带密钥伪随机函数 $PRF(\cdot)$；两个循环群 $\mathbb{G} = (\mathbb{Z}_p, +)$ 和 $\mathbb{H} = (\mathbb{Z}_q^*, \times)$；由 O 生成的初始向量（初始化向量）；

输出： S_b 输出 V^b，满足：(i) $V^0 \odot V^1$ 没有除"1"以外的相同元素，并且 (ii) 当且仅当所有 m 客户端都满足时，元素等于"1"将该元素编码为"1"（参见公式（6）；

```
/**
 *      聚合秘密份额
 */
1  V^b = Numpy.zeros((1, n))
2  对于 i = 0; i < n; i++ 执行
3  |   对于 j = 0; j < m; j++ 执行
4  |   |   V^b[i] := V^b[i] + V_j^b[i]
5  |   end
6  |   V^b[i] := V^b[i] − m^b % p
7  结束

/**
 *      编码聚合份额
 */
8  pos := 0
9  对每个 v ∈ V^b 执行
10 |   ct ← PRF_k(pos ⊕ IV)
11 |   g := (ct % (q − 2)) + 2                    // [2, q-1]
12 |   当 |g| ≠ p 时执行
13 |   |   g := ((g + 1) % (q − 2)) + 2 // g 生成元
14 |   end
15 |   V^b[pos] := g^v % q
16 |   pos++
17 结束

/**
 *      返回密文向量
 */
18 S_b 将 V^b 广播给 C_i, 0 ≤ i < m
```

复杂度。假设计算 $PRF(\cdot)$ 的成本为 c。第 2–7 行的复杂度为 O(nm)。第 9–17 行的复杂度为 O(nqc)。因此，算法1的总体计算复杂度为 O(nm+nqc)。消息总数很容易得出：第 18 行

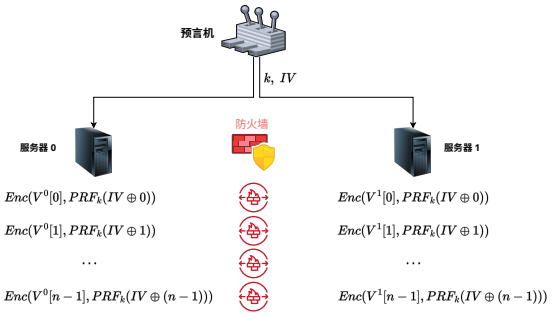


图2：Kaleido PSI 中由两个互不串通的服务器生成的随机密文

产生 m 条消息。由于有两台服务器，网络消息总数为 $2m$。通信轮次总数为 1，同样由第 18 行产生（假设 广播 是通过异步原语实现的）。

4.2 Kaleido 的正确性

我们需要证明 $U[i] = 1, 0 \le i < n$ 当且仅当 $V_j[i] = 1$，$0 \le j < m$。

4.2.1 $V_j[i] = 1 \implies U[i] = 1$. 如果 $V_j[i] = 1$，回想一下，这意味着第 j 个客户端持有属性 A_c 的第 i 个位置值。令 g_i 表示提升基的第 i 个，如算法1第 12 行所示。如 §4.1 所述，g_i 对 S_0 和 S_1 均相同。根据等式 (3)，有

$$\begin{aligned} U[i] &\equiv U^0[i] \times U^1[i] \pmod q \\ &= g_i^{V^0[i]} \times g_i^{V^1[i]} \\ &\equiv g_i^{V^0[i]+V^1[i] \pmod p} \\ &= g_i^{-m^0-m^1+\sum_{j=0}^{m-1} V_j[i]} \\ &= g_i^{-m+\sum_{j=0}^{m-1} 1} \\ &= g_i^0 \\ &= 1. \end{aligned} \tag{8}$$

4.2.2 $U[i] = 1 \implies V_j[i] = 1$. 如果 $U[i] = 1$，那么根据等式 (8)，必须满足以下等式：

$$-m^0 - m^1 + \sum_{j=0}^{m-1} V_j[i] = 0,$$

或者等价地：

$$\sum_{j=0}^{m-1} V_j[i] = m.$$

回想一下，$V_j[i] \in \{0, 1\}$，参见公式（6）。因此，满足上述等式的唯一方式是 $V_j[i] = 1, 0 \le j < m$，这正是我们所需要的。

4.3 Kaleido的可证明安全性

我们声称，由算法 1 支持的隐私集合求交方案是选择明文攻击下的不可区分性安全的，但所有客户端共享的属性值除外。这意味着，不允许对手查询服务器，使其返回所有客户端共享的某个属性值的密文。我们将上述内容表述为以下命题。

命题1。 如果算法*1*中的 $PRF(\cdot)$ 函数与理想随机数生成器不可区分，那么算法*1*中由 V^b 生成的 U 元素 $b \in \{0, 1\}$ 在选择明文攻击下的不可区分安全模型中是可证明安全的。

证明。设 A 表示能够攻破算法1安全性的对手。我们用 $MOD^{CPA}_{\mathcal{A},\ Kaleido}$ 表示多所有者数据库（MOD）中的隐私集合求交（PSI）实验：

(1) A 输出两个属性值 x_0 和 x_1，它们属于 $\mathbb{G}$，且至少有一个客户端不持有 x_0 或 x_1；(2) Kaleido服务器就一个值 $b \in \{0,1\}$ 达成一致，运行算法1，并向 A 发送 $Enc(x_b)$；(3) A 输出 $b' \in \{0,1\}$；(4) 若 $b' = b$，则实验输出定义为1，否则为0。若实验输出为1，我们记为 $MOD^{CPA}_{\mathcal{A},\ Kaleido} = 1$，并称 A成功。

令 g_0 和 g_1 分别表示 x_0 和 x_1的基，如算法1第12行所指定。我们考虑两种情况：(i)$g_0 \neq g_1$ 和(ii) $g_0 = g_1$。

- If $g_0 \neq g_1$, we have

$$Pr\left[MOD^{CPA}_{\mathcal{A},\ Kaleido} = 1 \mid g_0 \neq g_1\right] = \frac{1}{2},$$

因为 A 能做的最好也只是随机猜测 b。根据初等概率论，我们还知道以下结论成立：

$$Pr\left[MOD^{CPA}_{\mathcal{A},\ Kaleido} = 1 \wedge g_0 \neq g_1\right] \leq \frac{1}{2}. \tag{9}$$

- 若 $g_0 = g_1$，由于 $PRF(\cdot)$与随机生成元不可区分，因此以下等式成立：

$$Pr\left[g_0 = g_1\right] = \frac{1}{2^\lambda}.$$

由此可得

$$Pr\left[MOD^{CPA}_{\mathcal{A},\ Kaleido} = 1 \wedge g_0 = g_1\right] \leq \frac{1}{2^\lambda}. \tag{10}$$

结合等式(9)和(10)，可得

$$Pr\left[MOD^{CPA}_{\mathcal{A},\ Kaleido} = 1\right] \leq \frac{1}{2} + \frac{1}{2^\lambda} = \frac{1}{2} + negl(\lambda),$$

这意味着 A 成功的概率不会显著高于随机猜测，从而完成证明。□

4.4 Kaleido 案例研究

我们重新审视 §3.3 中讨论的示例，以展示所提出的 Kaleido PSI 的有效性。回想一下，$g_i = g_j, i \neq j$, 成立的概率可以忽略不计。假设 g_i等于 $i + 2\ \%\ 11$。例如， $U^0[3]$ 计算如下：

$$U^0[3] = (3 + 2\ \%\ 11)^{V^0[3]}\ \%\ 11 = 5^3\ \%\ 11 = 4,$$

并且 $U^1[3]$ 计算为

$$U^1[3] = (3 + 2\ \%\ 11)^{V^1[3]}\ \%\ 11 = 5^2\ \%\ 11 = 3.$$

根据上述算法， U^b 可计算为：

$$\begin{cases} U^0 = (1, 9, 1, 4, 9) \\ U^1 = (4, 9, 5, 3, 1) \end{cases}$$

因此，Kaleido PSI U 为：

$$U = (4, 4, 5, 1, 9).$$

我们可以看到， $U[3] = 1$ 仍然表明 A_c的第四个位置值由所有客户端共享，这正是我们所希望的。然而，其他值之间的相等或不等关系并不能揭示共享这些值的客户端数量。例如，尽管 $U[0] = U[1] = 4$，这并不意味着整数“0”和“1”由相同数量的客户端共享；事实上，只有 C_0 持有“0”，而除C_2之外的所有客户端都持有“1”。类似地，尽管 $U[0] \neq U[2] = 5$，这并不意味着整数“0”和“2”不是由相同数量的客户端共享——实际上，两者确实由相同数量的客户端共享（一个客户端 C_0 持有“0”，另一个客户端 C_3 持有“2”）。

5 评估

5.1 系统实现

我们使用约 1,000 行 Python 代码和 Bash 脚本实现了所提出的 Kaleido 方案，代码将发布在 https://github.com/。我们选择轻量级的 SQLite 作为本地数据库实例。请注意，SQLite 是基于文件的数据库，不支持网络访问。因此，我们通过 *paramiko* 库在远程 SQLite 实例之间实现了通信层，以支持安全数据传输和远程查询调用。一些最重要的库和依赖项包括：*python* 3.8.0、*sqlite* 3.31.1、*numpy* 2.21.0、*paramiko* 2.12.0、*scp* 0.14.4，以及密码学 39.9.0。

5.2 实验设置

5.2.1 测试床。我们在 Cloud- Lab [9] 托管的一个 34 节点集群上，单独部署 Kaleido 和其他基线方案，并搭配 SQLite [16] 使用。每个节点配备两颗 32 核 Intel Xeon Gold 6142 中央处理器、384 GB ECC DDR4-2666 内存和两块 1 TB 固态硬盘。操作系统映像为 Ubuntu 20.04.3 LTS，页面大小为 4 KB。所有服务器均通过

1 吉比特/秒控制链路（Dell D3048交换机）和 10 吉比特/秒实验链路（Dell S5048交换机）连接。我们仅使用实验链路进行评估。

具体而言，我们将集群中的34个节点命名为*node0–node33*。两台服务器分别运行在*node0* 和*node1*上，数据库所有者（即客户端）则部署在*node2–node33*上。为便于通信，我们为所有34个节点启用了免密码SSH连接，因为我们的评估侧重于性能指标，而非安全测量。

5.2.2 对比系统。对于所有参与对比的系统，加法群和乘法群的阶分别为113和227。

Prism。Prism [30] 在服务器编码期间分配一个静态生成元。如前文所述，Prism可能会泄露隐私集合求交之外的信息，例如哪些项目被相同数量的客户端共享。不过，由于生成元是固定的，Prism无需额外开销来随机化密文，因此其性能优于本文提出的各方案。

Kaleido-RND（朴素随机性）。在 Prism 中解决静态生成元问题的一种直接方法是随机选取一个生成元，我们将其称为*Kaleido-RND*。这要求两台服务器同步随机种子，因此会产生服务器间通信，这与 PSI 服务器不应相互串通的假设相悖。此外，我们尚不了解有什么有效的模型可以从形式上证明该方案的安全性。

Kaleido-AES（可证明安全）。我们选择 AES-128 [35]在 Kaleido原型系统中实现伪随机函数。使用 PKCS7 对明文数据进行填充，并采用 AES密码的密码分组链接（CBC）模式。初始向量（IV）设为 1234。

5.2.3 工作负载。

TPC-H 基准测试。合成数据集采用 TPC-H 3.0 版 0 [48], 这一标准数据库基准测试。TPC-H 共包含八张表；在我们的评估中，我们选取了 LineItem表——与 Pris m 中使用的表相同 [30], ，其中包含 6,001,215 个元组。我们关注的属性是*L_ORDERKEY*，它包含 1,500,000 个不同的值。每个数据库所有者，即客户端，都会从*L_ORDERKEY* 属性中随机选取 5,000,000 个值，而隐私集合求交的目标是找出各客户端集合共有的*L_ORDERKEY* 值。

*COVID-19*数据集。2020年4月到3月 2021 [8]的美国全国新冠肺炎统计数据。该数据集包含341天的数据和16项指标，例如死亡人数增加、阳性病例增加和住院人数增加。每个客户端随机选择300天的数据作为其本地数据。

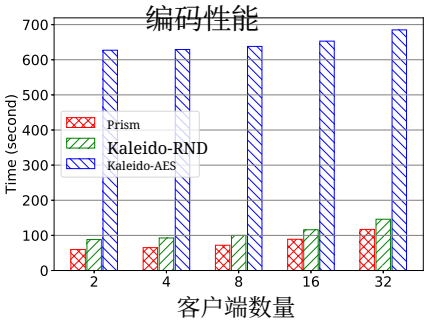


图 3：各种方案的编码性能

比特币交易历史。第二个应用是自2013年2月比特币首次在公开市场交易以来的交易量历史 [4] 。该数据包含从2013年2月至2022年1月按3天为周期累计的比特币交易量，共计1,086个大数值。每个客户端从其本地数据库中随机选择1,000个交易数据点。

人类基因*#38*序列。 第三个应用是人类基因组参考数据 38 [18],，通常称为 *hg38*，其中包含 34,424 行单一属性，例如转录位置、编码区域和外显子数量，最近一次更新于 2020年3月。每个客户端从其本地数据集中随机选择 30,000 行。

5.3 性能

*5.3.1 TPC-H*基准测试。

服务器编码。图 3 展示了各种方案的编码性能。我们比较了三种方案在服务不同数量客户端（从 2 到 32 个）时的服务器处理时间。不出所料，由于采用固定生成元 g，Prism 的性能最高。Kaleido-RND 比 Prism 稍慢，这是因为两台服务器之间需要通信，而且两台服务器都要承担随机化成本。Kaleido-AES 在编码 500 万个元组时会增加约 500 秒的开销；这是确保多所有者数据库上的隐私集合求交具备语义安全性的代价。请注意，这种开销属于计算成本，可通过并行处理降低，但本工作尚未探索这一方法。

客户端计算。在我们的实现中，客户端的工作可分为四个阶段：(i) 从本地 SQLite数据库加载数据，(ii) 对原始值进行哈希并将其映射为二进制标志，(iii) 将二进制标志拆分为两个加法秘密份额，以及 (iv) 从服务器返回的两个编码向量中恢复隐私集合求交结果。表 1 报告了上述四个阶段各自的耗时。我们观察到

表 1：客户端计算时间（秒）

基准测试	Load	Hash	拆分	恢复
TPC-H LineItem	5.615 秒	2.682 秒	9.358 秒	0.139 秒

表2：I/O成本（单位：兆字节）

基准测试	客户端向量	服务器向量
TPC-H LineItem	15 MB	17 MB

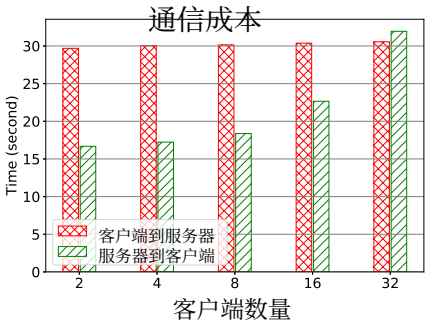


图4：不同客户端数量下的通信时间

最耗时的阶段是拆分，耗时近 10 秒。同样，由于向量中的元素彼此独立，因此可以利用多个 CPU核心并行处理；我们将在未来的工作中探讨这一点。

*I/O*成本。 表2报告了客户端和服务器的本地 I/O 成本。请注意，这些数值针对单个客户端或服务器，例如，每个客户端持久化存储一个大小为 15 MB 的向量——因此，整个系统的 I/O 开销总计为 $15\times m$ MB，其中 m 表示数据库所有者（客户端）的数量。

通信成本。 图4报告了不同客户端数量下的通信耗时。我们观察到，随着客户端数量的扩展，上行通信耗时几乎保持不变；然而，参与的客户端越多，下行流量耗时就越长。当客户端数量足够多时，例如 32 个，服务器到客户端的通信耗时会超过反方向的通信耗时。

5.3.2 应用。受篇幅限制，我们仅报告这三种应用在客户端数量为 2、4 和 8 时的性能。我们观察到，在这些实际应用中，性能开销主要来自通信：客户端和服务器的大部分时间都花在发送向量上，耗时比其他阶段高出几个数量级。

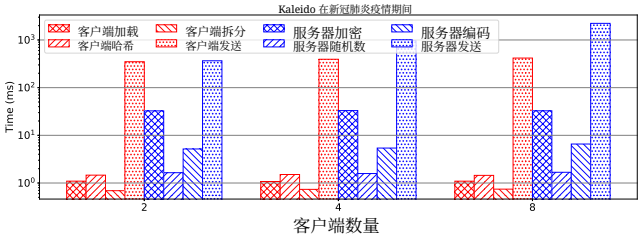


图5：Kaleido 在新冠肺炎患者记录上的性能 [8]

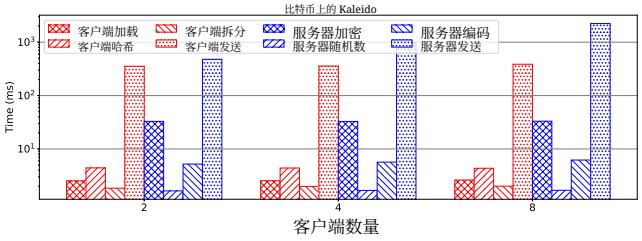


图6：Kaleido 在比特币交易历史上的性能 [4]

新冠肺炎。图5报告了 Kaleido 和 Prism 在新冠肺炎数据上的分阶段性能。对于客户端，我们观察到，客户端到服务器的通信占据了主要成本。对于服务器，同样是服务器到客户端的通信占据了总体成本的大部分，而且随着客户端数量增加，这一比例也会上升。因此，尽管从计算角度来看，Kaleido 引入的密码学成本相当显著，但整个系统的真正瓶颈在于通信。事实上，密码学组件的计算成本还可以进一步降低（本工作未对此进行探讨）。

比特币。 图6报告了比特币交易历史的性能。我们观察到与新冠肺炎数据相似的模式。更明显的是，数值表明服务器通信占据服务器端的总体成本主导地位：仅有两个客户端时，服务器的通信耗时是密码学组件（即 AES [35]）的 10 倍。这些结果再次证明了所提出的 Kaleido 方案具有实用性：尽管其成本显著高于朴素的随机方法，但仍比通信耗时低一个数量级。

人类基因*#38*序列。 图 7 展示了在更大型数据集上的性能：超过 30,000 条人类基因#38序列。我们观察到，服务器的性能与前两个数据集大致相同；然而，

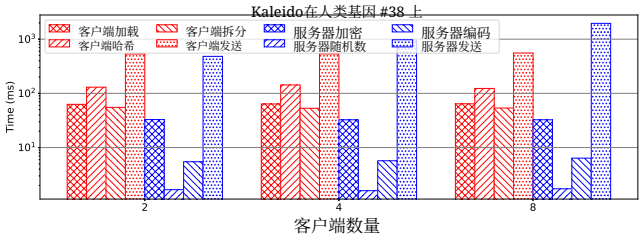


图 7：Kaleido 在人类基因 #38 序列上的性能 [18]

这一次客户端的负载更重。这可以理解，因为在这种情况下，客户端必须处理与其本地数据库相关的更多数据项。

6 相关工作

本节回顾了另外三种用于确保外包数据库机密性的重要技术。这些技术都与基于秘密共享的方案（例如 Prism 和 Kaleido）正交。

6.1 加密存储

云服务商提供的数据库实例被视为加密数据的存储，而客户端负责执行复杂查询。只有在以下条件成立时，这种方案才可行：(i) 查询涉及的关系足够小，传输这些关系所产生的网络开销可以接受；(ii) 用户有能力（包括计算和存储能力）在本地执行查询。我们强调，这种方案可能会使将数据库服务外包给云端失去意义。

6.2 加密元组

原始关系 R 中的每个元组都被加密为一个密文，并存储在新关系 R^s的列 T中。对于 R中的每个属性 A_i，在 R^s中都有一个对应的属性 A_i^s，其值为 $R.A_i$的索引。该索引通常根据某些分区标准分配为一个随机整数，并可通过存储在客户端（即用户的本地节点）上的元数据获取。因此，云服务提供商存储的模式为 $R^s(T, A_1^s, \ldots, A_i^s, \ldots)$。用户提交查询 Q时，客户端会将 Q 拆分为两个子查询 Q_s 和 Q_c。 Q_s 充当过滤器，根据 R^s 中的索引排除不符合条件的元组，并将符合条件的元组（以密文形式）传输给客户端。 Q_c 随后确保使用据称存储在客户端上的密钥解密加密元组后，排除这些假阳性元组。这种方法涉及客户端（即用户）和服务器（即云服务提供商）完成查询时常用的一类方法，称为信息隐藏 方法 [14]。

6.3 加密字段

第三种方法旨在尽量减少客户端在处理云服务商存储的加密数据查询时的参与。其思路是以更细的粒度加密关系——分别加密关系中的每个属性。这种方法的关键挑战在于其表达能力，例如，如何对加密字段执行算术运算或字符串运算。虽然全同态加密（FHE） [12] 能够支持大量计算问题，但目前全同态加密的实现性能尚无法满足实际数据库系统 [43, 44]的要求。一种替代方案是部分同态加密（PHE）方案 [10, 37],，其速度比全同态加密快几个数量级，但仅支持单一代数运算。传统的部分同态加密方案面向公钥（非对称）加密而设计，这有利于通过不安全信道直接分发密钥，但其成本远高于密钥（对称）加密。然而，在外包数据库的场景中，用户通常既是发送方也是接收方，因此无需分发密钥。为此，人们提出了对称（部分）同态加密方案 [38, 45]。

7 结论与未来工作

本文首先展示了当前最先进的隐私集合求交方案 Prism 在外包数据库中的一项安全漏洞。随后，本文提出了一系列名为 Kaleido 的新原语，以扩展 Prism，使新方案具备语义安全性。本文对正确性和安全性均进行了形式化证明，并通过一个贯穿全文的运行示例阐释其直观原理。Kaleido 系统原型已实现，并部署在 CloudLab [9] 上，包含两台服务器和 32 个数据库所有者。针对 TPC-H 基准和三个实际应用开展的大量实验，证实了所提出的 Kaleido 方案行之有效。

我们的未来工作包括两个方面。首先，我们将利用底层的多个/众多 CPU 核心，在 Kaleido 中并行执行开销较大的密码学运算。挑战在于，如何确保多个（不合谋）服务器彼此不通信，同时又能就随机化的群生成元保持同步。其次，我们将采用一些压缩方案来缩减客户端和服务器生成的向量，从而降低通信成本。第二个研究方向的挑战在于权衡向量压缩和解压缩所需的计算成本，以及压缩向量带来的通信节省。

参考文献

[1] Diego F. Aranha、Anders P. K. Dalskov、Daniel Escudero 和 Claudio Orlandi。基于 LSS 同构的改进门限签名、主动秘密共享和输入认证。载于 Patrick Longa 和 Carla Ràfols 编，密码学进展——*LATINCRYPT 2021*：第七届拉丁美洲密码学与信息安全国际会议，波哥大，哥伦比亚，*2021*年*10*月*6*日至*8*日，论文集，《计算机科学讲义》第 12912 卷，第 382–404 页。施普林格，2021年。[2] Saikrishna Badrinarayanan、Sourav Das、Gayathri Garimella、Srinivasan Raghuraman 和 Peter Rindal。利用聚合树实现多重性的秘密共享连接。载于 Heng Yin、Angelos Stavrou、Cas Cremers 和 Elaine Shi 编，*2022*年 *ACM SIGSAC* 计算机与通信安全会议论文集（*CCS 2022*），美国加利福尼亚州洛杉矶，*2022*年*11*月*7*日至*11*日，第 209–222 页。ACM，2022年。[3] D. Beaver、S. Micali 和 P. Rogaway。安全协议的轮复杂度。载于第二十二届 *ACM* 年度计算理论研讨会论文集，STOC '90，第 503–513 页，美国纽约州纽约市，1990年。计算机协会。[4] 比特币交易历史。https://www.blockchain.com/charts/trade-volume，访问于2023年。[5] Miguel Castro 和 Barbara Liskov。实用拜占庭容错与主动恢复。*ACM* 计算机系统汇刊，20(4):398–461，2002年11月。[6] Hao Chen、Kim Laine 和 Peter Rindal。基于同态加密的快速隐私集合求交。载于*2017*年 *ACM SIGSAC* 计算机与通信安全会议论文集，CCS '17，第 1243–1255 页，美国纽约州纽约市，2017年。计算机协会。[7] Jung Hee Cheon、Stanislaw Jarecki 和 Jae Hong Seo。准线性复杂度的多方隐私保护集合求交。*IEICE* 电子、通信与计算机科学基础汇刊，95(8):1366–1378，2012年1月。[8] 新冠肺炎数据。https://covidtracking.com/data/download/national-history.csv，访问于2023年。[9] Dmitry Duplyakin、Robert Ricci、Aleksander Maricq、Gary Wong、Jonathon Duerig、Eric Eide、Leigh Stoller、Mike Hibler、David Johnson、Kirk Webb、Aditya Akella、Kuangching Wang、Glenn Ricart、Larry Landweber、Chip Elliott、Michael Zink、Emmanuel Cecchet、Snigdhaswin Kar 和 Prabodh Mishra。CloudLab 的设计与运行。载于*USENIX* 年度技术会议（*ATC*）论文集，第 1–14 页，2019年7月。[10] T. Elgamal。基于离散对数的公钥密码系统和签名方案。*IEEE* 信息论汇刊，31(4):469–472，1985年。[11] Michael J. Freedman、Kobbi Nissim 和 Benny Pinkas。高效的私有匹配与隐私集合求交。载于 Christian Cachin 和 Jan Camenisch 编，密码学进展——*EUROCRYPT 2004*：密码技术理论与应用国际会议，瑞士因特拉肯，*2004*年*5*月*2*日至*6*日，论文集，《计算机科学讲义》第 3027 卷，第 1–19 页。施普林格，2004年。[12] Craig Gentry。使用理想格的全同态加密。载于第四十一届 *ACM* 年度计算理论研讨会（*STOC*）论文集，2009年。[13] O. Goldreich、S. Micali 和 A. Wigderson。如何玩任何心理游戏。载于第十九届 *ACM* 年度计算理论研讨会论文集，STOC '87，第 218–229 页，美国纽约州纽约市，1987年。计算机协会。[14] Hakan Hacigümüş、Bala Iyer、Chen Li 和 Sharad Mehrotra。在数据库服务提供商模型中对加密数据执行 SQL。载于*2002*年 *ACM SIGMOD* 国际会议论文集数据管理，SIGMOD '02，第216–227页，纽约，纽约州，美国，2002年。计算机协会。[15] Carmit Hazay 和 Muthuramakrishnan Venkitasubramaniam。可扩展多方私有集合求交。载于 Serge Fehr 编，公钥密码学——*PKC 2017*——第*20*届 *IACR*公钥密码学实践与理论国际会议，阿姆斯特丹，荷兰，*2017*年*3*月*28*日至*31*日，论文集，第*I*部分，《计算机科学讲义》第10174卷，第175–203页。施普林格，2017年。[16] Richard D Hipp。SQLite，2020年。[17] Yan Huang、David Evans 和 Jonathan Katz。隐私集合求交：混淆电路是否优于定制协议？载于第*19*届年度网络与分布式系统安全研讨会（*NDSS 2012*），圣迭戈，加利福尼亚州，美国，*2012*年*2*月*5*日至*8*日。互联网协会，2012年。[18] 人类基因组数据库。http://hgdownload.soe.ucsc.edu/ goldenPath/hg38/database/，访问于2023年。[19] Roi Inbar、Eran Omri 和 Benny Pinkas。通过混淆布隆过滤器实现高效、可扩展的多方隐私集合求交。载于 Dario Catalano 和 Roberto De Prisco 编，网络安全与密码学——第*11*届国际会议（*SCN 2018*），阿马尔菲，意大利，*2018*年*9*月*5*日至*7*日，论文集，《计算机科学讲义》第11035卷，第235–252页。施普林格，2018年。[20] Mihaela Ion、Ben Kreuter、Ahmet Erhan Nergiz、Sarvar Patel、Shob- hit Saxena、Karn Seth、Mariana Raykova、David Shanahan 和 Moti Yung。关于部署安全计算：具有基数计算的私有交集求和。载于*IEEE*欧洲安全与隐私研讨会（*EuroS&P 2020*），热那亚，意大利，*2020*年*9*月*7*日至*11*日，第370–389页。IEEE，2020年。[21] Mitsuru Ito、Akira Saito 和 Takao Nishizeki。实现一般访问结构的秘密共享方案。《日本电子与通信》（第三部分：基础电子科学），72(9):56–64，1989年。[22] Don Johnson、Alfred Menezes 和 Scott A. Vanstone。椭圆曲线数字签名算法（ECDSA）。《国际信息安全杂志》，1 63，2001 1 36( ): –。[23] Seny Kamara、Payman Mohassel、Mariana Raykova 和 Seyed Saeed Sadeghian。将隐私集合求交扩展至十亿元素集合。载于 Nicolas Christin 和 Reihaneh Safavi-Naini 编，金融密码学与数据安全——第*18*届国际会议（*FC 2014*），基督堂，巴巴多斯，*2014*年*3*月*3*日至*7*日，修订精选论文，《计算机科学讲义》第8437卷，第195–215页。施普林格，2014年。[24] Florian Kerschbaum。抗串谋的隐私集合求交外包。载于第*27*届*ACM*年度应用计算研讨会论文集，SAC '12，第1451–1456页，纽约，纽约州，美国，2012年。计算机协会。[25] Lea Kissner 和 Dawn Song。隐私保护集合运算。载于第*25*届密码学进展国际年会论文集，CRYPTO'05，第241–257页，柏林，海德堡，2005年。施普林格出版社。[26] Vladimir Kolesnikov。门评估秘密共享与安全的单轮两方计算。载于 Bimal Roy 编，密码学进展——*ASIACRYPT 2005*，第136–155页，柏林，海德堡，2005年。施普林格柏林海德堡。[27] Vladimir Kolesnikov、Ranjit Kumaresan、Mike Rosulek 和 Ni Trieu。高效批处理不经意伪随机函数及其在隐私集合求交中的应用。载于*2016*年*ACM SIGSAC*计算机与通信安全会议论文集，CCS '16，第818–829页，纽约，纽约州，美国，2016年。计算机协会。[28] Vladimir Kolesnikov、Naor Matania、Benny Pinkas、Mike Rosulek 和 Ni Trieu。基于对称密钥技术的实用多方隐私集合求交。载于*2017*年 *ACM SIGSAC* 计算机与通信安全会议论文集，CCS '17，第1257–1272页，纽约，纽约州，美国，2017年。计算机协会。

[29] Phi Hung Le、Samuel Ranellucci 和 S. Dov Gordon。借助不可信第三方的两方隐私集合求交。载于*2019*年 *ACM SIGSAC* 计算机与通信安全会议论文集，CCS '19，第2403–2420页，纽约，纽约州，美国，2019年。计算机协会。[30] Yin Li、Dhrubajyoti Ghosh、Peeyush Gupta、Sharad Mehrotra、Nisha Panwar 和 Shantanu Sharma。Prism：多所有者外包数据库上的私有可验证集合计算。载于*2021*年国际数据管理会议论文集，SIGMOD '21，第1116–1128页。计算机协会，2021年。[31] Yehuda Lindell。如何进行模拟——模拟证明技术教程，第277–346页。施普林格国际出版公司，查姆，2017年。[32] Yehuda Lindell。安全多方计算。《*ACM*通讯》，64(1):86–96，2020年12月。[33] Payman Mohassel、Peter Rindal 和 Mike Rosulek。秘密共享数据的快速数据库连接与隐私集合求交。载于 Jay Ligatti、Xinming Ou、Jonathan Katz 和 Giovanni Vigna 编，*CCS '20*：*2020*年*ACM SIGSAC*计算机与通信安全会议，线上活动，美国，*2020*年*11*月*9*日至*13*日，第1271–1287页。ACM，2020年。[34] Moni Naor、Benny Pinkas 和 Reuban Sumner。隐私保护拍卖与机制设计。载于第*1*届*ACM*电子商务会议论文集，EC '99，第129–139页，纽约，纽约州，美国，1999年。计算机协会。[35] 国家标准与技术研究院。高级加密标准。*NIST FIPS PUB 197*，2001年。[36] OpenSSL。https://github.com/openssl/openssl，访问于2023年。[37] Pascal Paillier。基于合数阶剩余类的公钥密码系统。载于第*17*届密码技术理论与应用国际会议论文集，EURO- CRYPT'99，第223–238页，柏林，海德堡，1999年。施普林格出版社。[38] Antonis Papadimitriou、Ranjita Bhagwan、Nishanth Chandran、Ramachandran Ramjee、Andreas Haeberlen、Harmeet Singh、Abhishek Modi 和 Saikrishna Badrinarayanan。利用 seabed 对加密数据集进行大数据分析。载于第*12*届*USENIX*操作系统设计与实现会议（*OSDI*）论文集，第587–602页，美国，2016年。USENIX协会。[39] Benny Pinkas、Mike Rosulek、Ni Trieu 和 Avishay Yanai。基于Paxos的隐私集合求交：快速的恶意安全隐私集合求交。载于 Anne Canteaut 和 Yuval Ishai 编，密码学进展——*EUROCRYPT 2020*——第*39*届密码技术理论与应用国际年会，萨格勒布，克罗地亚，*2020*年*5*月*10*日至*14*日，论文集，第*II*部分，《计算机科学讲义》第12106卷，第739–767页。施普林格，2020年。[40] Benny Pinkas、Thomas Schneider、Gil Segev 和 Michael Zohner。Phasing：使用基于置换的哈希进行隐私集合求交。载于第*24*届*USENIX*安全研讨会论文集，SEC'15，第515–530页。USENIX协会，2015年。[41] Benny Pinkas、Thomas Schneider、Christian Weinert 和 Udi Wieder。通过布谷鸟哈希实现高效的基于电路的隐私集合求交。载于 Jesper Buus Nielsen 和 Vincent Rijmen 编，密码学进展——*EUROCRYPT 2018*——第*37*届密码技术理论与应用国际年会，特拉维夫，以色列，*2018*年*4*月*29*日至*5*月*3*日，论文集，第*III*部分，《计算机科学讲义》第10822卷，第125–157页。施普林格，2018年。[42] Benny Pinkas、Thomas Schneider 和 Michael Zohner。基于不经意传输扩展的可扩展隐私集合求交。《*ACM*隐私与安全汇刊》，21(2):7:1–7:35，2018年。[43] Rishabh Poddar、Tobias Boelter 和 Raluca Ada Popa。Arx：采用语义安全加密的加密数据库。《*VLDB*基金会会刊》，12(11):1664–1678，2019年。

[44] Raluca Ada Popa、Catherine Redfield、Nickolai Zeldovich 和 Hari Balakrishnan。《Cryptdb：通过加密查询处理保护机密性》。发表于*ACM*操作系统原理第二十三届研讨会（*SOSP*）论文集，2011年。[45] Savvas Savvides、Darshika Khandelwal 和 Patrick Eugster。《对称加密数据集上的高效保密性保护数据分析》。《VLDB基金会会刊》，13(8):1290–1303，2020年4月。[46] Adi Shamir。《如何分享秘密》。《ACM通讯》，22(11):612–613，1979年11月。[47] Adi Shamir。《如何分享秘密》。《ACM通讯》，22(11):612–613，1979年11月。[48] TPC-H 3.0.0。http://tpc.org/tpc_documents_current_versions/current_specifications5.asp，访问于2023年。[49] Andrew C. Yao。《安全计算协议》。发表于第*23*届年度计算机科学基础研讨会，第160–164页，1982年。[50] Samee Zahur、Mike Rosulek 和 David Evans。《两个半门合成一个整体——使用半门减少混淆电路中的数据传输》。发表于 Elisabeth Oswald 和 Marc Fischlin 编，《密码学进展——EUROCRYPT 2015——第34届密码技术理论与应用国际会议，保加利亚索非亚，2015年4月26日至30日，论文集，第二部分》，《计算机科学讲义》第9057卷，第220–250页。施普林格，2015年。