多方安全计算热点：隐私保护集合求交技术 (PSI) 分析研究报告

崔泓睿 * 刘天怡 * 郁昱 * 程越强 ‡ 张煜龙 ‡ 韦韬 ‡
* 上海交通大学 LATTICE 实验室
‡ 百度安全实验室

2019 年 8 月 14 日

摘要

隐私保护集合交集 (Private Set Intersection, PSI) 计算属于安全多方计算领域的特定应用问题，不仅具有重要的理论意义也具有很强的应用价值。随着用户数据的隐私保护越来越受到重视，这一方向的研究更是符合人们日益强烈的在享受各类依赖个人信息的业务的便利性的同时最大程度保护个人信息私密性的需要。本文首先分析比较了 17 种基于多方安全计算或者全同态加密的 PSI 协议方案，包括攻击模型、安全模型、性能测试等。结果表明目前 EC-ROM/DE-ROM [55] 是目前最快的基于密码学的公开安全 PSI 协议。同时，文章也对最新的基于 SGX 的 PSI 协议做了对比分析。结果表明百度安全实验室自主提出的基于 SGX 的 PSI 协议在性能方面是目前最快的 PSI 协议 (EC-ROM 和 DE-ROM) 的 60 倍。而且在安全性，灵活性和多功能性等方面 SGX PSI 比传统的 PSI 具有不同程度的优势，比如 SGX PSI 可以自适应 2 方，3 方以及任意不同数量的参与者，而传统 PSI 在参与方数目自适应方面尚有明显差距。

1 引言

作为安全多方计算领域具有广泛的应用场景的一类协议，隐私保护集合交集技术在近年来得到了极大的优化，达到了在某些场景下与目前正在使用的非安全交集技术同一量级的运行复杂度（从计算和通信意义上）。在这一节我会将隐私保护集合交集的协议目标进行阐述，并向读者展示其可能的应用场景。

1.1 问题描述

隐私保护集合交集协议允许持有各自集合的两方来共同计算两个集合的交集运算。在协议交互的最后，一方或是两方应该得到正确的交集，而且不会得到交集以外另一方集合中的任何信息。保护集合的隐私性是在很多场景下是自然甚至是必要的需求，比如当集合是某用户的通讯录或是某基因诊断服务用户的基因组，这样的输入就一定要通过密码学的手段进行保护。

值得注意的虽然 PSI 协议的发展非常迅速，而且对于数据隐私性保护的需求也日益强烈，在目前很多应用场景下，高效而不安全的协议还是主流选择。因此了解 PSI 协议的最新成果以及它们合适的应用场景将会对使用 PSI 协议替换现有非安全协议起到非常大的帮助。

1.2 PSI 的应用场景

PSI 拥有很多实际应用场景，这里只挑选典型的两种应用场景—计算广告的实际效果和寻找联系人。

计算广告的实际效果 线上广告是一种重要的广告形式。对于广告的有效程度的衡量的常见方法是计算所谓的转换率，也就是浏览广告的用户中有多少用户最终浏览了相应的商品页面，或是最终购买了相应的商品或是服务。一种通用的计算方法是由计算浏览广告的用户信息（由广告发送方占有）和完成相应交易的用户信息（由商家占有）的交集来计算（如计算交易总额或是总交易量等）。而与此同时双方的用户信息又是私密的，如果使用不安全的协议会导致一方的信息暴露给另一方，从而造成用户和商家或是广告主的隐私泄露。

注意到现有的很多类似应用场景下，所使用的协议仍然是不安全的，也就是说在计算广告相关数据的同时，隐私信息也受到了严重的威胁。[1]

寻找联系人 当一个用户注册使用一种新的服务（如微信、Whatsapp 等）的时候，从用户的现有联系人中寻找有哪些已经注册了同类的服务是一种在大多数情况下十分必要的操作。通过将用户的联系人发送给服务提供商可以有效地完成这项功能，但是与此同时用户的联系人信息，一种在大多数情况下被认为是隐私的信息，也被暴露给服务提供商了。因此在这种场景下，将用户的联系人信息作为一方的输入，将服务提供商的所有用户信息作为另一方的输入来进行 PSI 协议可以完成发现联系人的功能，而且可以防止交集以外的信息泄露给任何一方。

值得注意的是在这种特定的应用场景下，大多数情况是用户的联系人集合的基数远小于已注册的用户集合的基数。传统的 PSI 协议考虑的是两方集合大小基本相等的情况，在这种场景下大集合将会成为性能瓶颈。一些研究成果集中对于这种比较常见而有实际应用价值的场景做优化，并且取得了较好的成果。

2 符号与定义

在这一节我会给出描述协议需要用到的一些基本符号，PSI 协议需要达到的目标，协议中考虑的两类敌手模型，在安全性证明中使用到的假设以及构建 PSI 协议中用到的基础协议。

2.1 符号说明

表1列举了后续协议说明与比较中使用到的符号，并附上了简单的描述。在集合大小不对等的 PSI 协议中，持有较大集合的一方通常被称作发送端（Sender），持有较小集合的一方被叫做接收端（Receiver），此时就满足 $N_X \gg N_Y$。这种名称也在集合大小对等的协议描述中作为标准名称使用。

符号	描述
S	服务端或是发送端
R	客户端或是接收端
X, Y	发送端和接收端的集合
N_X, N_Y	发送端和接收端集合的大小
m	哈希表的大小
v	消息编码的长度
σ	集合元素的长度
PKF	伪随机函数
κ, λ	计算意义的安全参数和统计意义的安全参数
ρ, ϕ	非对称安全参数与椭圆曲线的规模 [2]

表 1: 符号说明

2.2 理想的 PSI 协议

理想的 PSI 协议由一个发送方 S 和一个接收方 R 参与。其中发送方持有集合 X，大小为 N_X；接收方持有集合 Y，大小为 N_Y。集合元素是长度为 σ 的二进制字符串。在大多数情况下 N_X 和 N_Y 是公开的。（如果需要将这样的信息保密，可以通过补充“哑元”（dummy items）到一个固定大小的方式通过公开集合大小的协议实现交集功能。）理想的 PSI 协议会计算 X 和 Y 的交集，将 $X \cap Y$ 发送给接收者，不将任何信息发送给发送者。接收者不能获得 $X \setminus Y$ 中的任何信息，发送者也不能获得 $Y \setminus X$ 或是 Y 的任何信息。

[1]一个例子是 Facebook 与信用卡信息持有方的合作：`https://www.eff.org/deeplinks/2012/09/deep-dive-facebook-and-datalogix-whats-actually-getting-shared-and-how-you-can-opt.`

[2]根据 NIST 的推荐标准，当安全等级为 128 比特，使用 K-283 曲线的时候，$\rho = 3072$，$\phi = 284$。

2.3 攻击模型

为了证明某种协议的安全性，敌手的能力和安全性的含义是必须严格定义的。关于安全性的严格定义在不同协议中有着不同的体现，但是思想都基于2.2节中的理想 PSI 协议的功能。对于敌手定义，密码学中常见的的三种定义为[62]：

1. 半诚实模型 (honest but curious adversary, HbC)。协议的各参与方遵守协议的执行过程，但可以在协议执行过程中，根据输入和协议的输出信息推断其他参与者的信息。
2. 恶意模型（malicious adversary, Mal)。参与者不遵守协议的执行过程，可能拒绝参与协议、修改隐私的输入集合信息、提前终止协议的执行等，因此需要使用更多的密码协议或技术（位比特承诺协议、零知识证明等）来保证计算结果的正确性。
3. 隐蔽敌手模型（covert adversary)。是一种安全性介于半诚实模型和恶意模型之间的更符合真实场景的模型，由于担心恶意行为被协议检测出来并受到惩罚，隐蔽敌手使其恶意行为混淆在正常行为中，只能以一定的概率被检测到。

安全多方计算协议一般会存在半诚实模型下安全的版本和恶意模型下安全的版本。虽然半诚实模型对敌手的限制很大，在很多情况下并不是合理的假设，但是首先设计出半诚实模型可以作为设计恶意模型安全协议的第一步（GMW 编译器 [28] 可以实现从半诚实模型到恶意模型的通用、但不高效的转化）；其次在某些场景下，半诚实模型中敌手必须按照协议规定进行交互的限制是合理的（比如恶意行为一旦发现就有很严格的处罚的场景）；最后恶意模型为了保证安全性会给协议带来一些额外的负担，使得半诚实模型下安全的版本会比恶意模型安全的协议高效很多。

同样，在目前的 PSI 协议中，常见的敌手模型为半诚实模型和恶意模型。而且由于恶意模型中一方可能会刻意获取另一方的信息（通过主动地偏离协议的规定来达到这一目的），协议需要使用额外的手段来防止这类攻击的可能，因此恶意模型下安全的协议的复杂程度和开销一般都大于半诚实模型下安全的协议。

2.4 安全性假设

在协议的安全性证明过程中，有两类非常重要的基础假设或是基础模型，分别是标准模型和随机预言模型。

1. 标准模型（Standard Model, Std）指不依赖任何假想模型，仅依赖于被广泛接受的困难性假设（如整数分解、离散对数、格问题等），这些数学难题是攻击者在多项式时间内不可解的。仅使用困难性假设证明安全的机制称为在标准模型下是安全的。
2. 随机预言模型（Random Oracle Model, ROM）比标准模型多了一个随机预言机的假设，随机预言机假设存在一个公共的、随机选择的函数（理论上的黑盒子），只能通过问询的方式进行计算，对每一次查询都从输出域中均匀随机地输出一个响应，对相同的查询总是得到相同的响应。由于现实中没有函数能够实现真正的随机函数，因此随机预言模型下可证明安全的方案在实际应用中通常用 Hash 函数进行实例化。

与之前敌手模型类似，RO 模型由于引入了更多的假设，在 RO 模型下证明安全的协议通常需要加入额外的构造才能被证明是在标准模型下是安全的。值得注意的是为了降低使用假设的数量，一些协议的分析中也使用了一种“关联鲁棒性”（correlation robustness）的假设（比如在 [38, 50] 中），作为一种更弱的假设来替代随机预言假设。

2.5 基础协议

在这一节中我将介绍在不同 PSI 协议构造中使用到的密码学基础协议。

不经意传输（Oblivious Transfer） 不经意传输协议（OT）是基于公钥密码体制的密码学基本协议，是安全多方计算的基石。最基本的二选一（1-out-of-2）OT 协议的发送方 Bob 输入 2 个随机位串 (x_0, x_1)，接收方 Alice 输入选择向量 c；协议结束后 Alice 获得选择向量对应的位串 x_c，对另一个位串 x_{1-c} 一无所知；Bob 的输出为空 [21]。在 1998 年 Impagliazzo 和 Rudich 证明了从不经意传输协议到单向函数的黑盒式规约蕴含着另一个难以被证明的问题，即 $P \neq NP$ 的问题，并表示不存在黑盒方式的 OT 协议 [32]，因此 OT 协议通常是基于公钥密码体制来构造. 然而在安全多方计算应用中一般需要大量的 OT 实例，计算复杂的模指数运算，使得 OT 实例的实用价值不高。1996 年 Beaver 依据混合加密构想提出了第一个非黑盒方式的不经意传输扩展协议 [2]，可以执行少数基础 OT 协议（传统的基于公钥加密算法的 OT 协议）来构造大量的 OT 协议，然而 Beaver 提出的协议需要计算复杂的伪随机发生器，在实际中也不高效，但是扩展协议的思想具有重要的影响。基于 OT 扩展协议的思想 Ishai 等人在 2003 年提出了以黑盒方式构造的 OT 扩展协议 [33]，将基础 OT 协议和随机预言模型相结合，把少量基础 OT 的计算代

价通过对称加密操作均摊到大量的 OT 操作，可以达到一分钟执行数百万次的 OT 协议，该协议可以同时满足实用性和安全性需求，具有重要的意义并得到很广泛的应用，例如用于 GMW 协议、姚氏加密电路、PSI 等。随着人们对实用性要求越来越高，OT 扩展协议得到了快速发展，主要包括对 N 选一不经意传输扩展协议 [37] 及随机 OT 协议 [1] 的提出。

加密电路（Garbled Circuit） 混乱电路（GC）模型最早是由图灵奖获得者姚期智在 1986 年提出的半诚实模型下的姚氏电路 [60]，用来解决著名的百万富翁问题。姚氏电路主要是将任意功能函数转化为布尔电路，由 Alice 生成混乱电路表、Bob 计算混乱电路；针对每一个电路门进行两重对称加密运算，调用二选一 OT 协议进行混乱电路表中秘钥信息交换。早期的安全函数计算问题主要采用混乱电路来解决，由于混乱电路对每一位进行电路门计算并且电路门数量巨大，导致计算效率较低，例如计算 AES 加密大约需要 30000 个电路门 [35]，计算 50 个字符串的编辑距离大约需要 250000 个电路门。混乱电路作为通用的安全多方计算工具，可以用来计算任意的功能函数，相对于特定问题的安全协议计算效率较低。针对这些问题，研究者提出了一系列的电路优化策略 [63]，包括 Free-XOR、行约减 [49, 44]、Half-Gate 技术 [61]。此外还提出了新的混乱电路协议，包括由 Goldreich 等人提出基于秘密共享和 OT 协议的 GMW 编译器 [27] 以及基于剪切-选择技术（cut and choose）[47] 的适用于恶意模型的混乱电路等。除了对布尔电路的研究人们也提出了算术电路，完成有限域上加法或乘法运算。混乱电路方法作为安全多方计算问题的一般通用解决方法，近年来得到了快速的发展，已经有很多实用的安全多方计算工具，例如 Fairplay[42]，FastGC[30]，Oblivm[41]，SEPIA[7]，VIFF[15]，Sharemind[4] 等。

同态加密（Homomorphic Encryption） 同态加密（HE）是满足同态性质的公钥加密技术，属于语义安全的公钥加密体制范畴。同态加密对密文进行某种算术操作（加或者乘），满足对密文计算结果的解密值与对明文进行同样算术操作的值是相同的性质。由于计算是在密文上进行，因此同态加密是一种常用的实现隐私保护的方法。同态加密根据方案支持的操作种类可以分为支持部分同态操作（加法或者乘法）的加密方案与支持全部同态操作的加密方案。其中前一种的例子包括 Paillier 加密 [46]，支持加法同态运算以及 ElGamal 加密 [20]，支持乘法同态运算。全同态加密方案的例子有 [25, 6, 5, 26]。在安全多方计算领域一种通用的通过增加计算量来节省通信量的方法是通过使用加法同态的加密方案来实现乘法运算。通过这种方法可以节省不经意传输协议的使用次数，从而降低通信量。ABY[18] 与 Sharemind[4] 就是通过这种方法实现乘法的两种安全多方计算框架。

秘密共享（Secret Sharing） 秘密共享（SS）将秘密以适当的方式分为 n 份，每一份由不同的管理者持有，每个参与者无法单独恢复秘密，只有达到指定数目的参与者才能恢复秘密。构建秘密共享系统的关键是设计好的秘密拆分和恢复方式. 第一个秘密共享方案是 (t, n) 门限秘密共享方案，由 Shamir[57] 和 Blakley[3] 分别在 1979 年各自独立提出，他们的方案分别是基于拉格朗日插值法和线性几何投影性质设计的. 此后很多研究者提出了不同的秘密共享实现方法，如基于中国剩余定理的秘密共享策略、可验证的秘密共享等。秘密共享在密钥管理分布式数据安全领域有许多应用，如电子投票、密钥托管、电子支付协议等，可以防止密钥存储过于集中，是一种兼顾机密性和可靠性的方法。

3 协议分类

下面在这一节中，我将给出 PSI 协议的大体分类，其中包括基于 Hash 函数、不安全的交集协议，基于公钥加密体系的 PSI，基于混乱电路的 PSI 以及基于不经意传输协议扩展的 PSI。在每一类中我会阐述相关协议的大体内容、优化过程以及适合的应用场景。

3.1 不安全的交集协议

最基本的交集协议是协议双方 S, R 都在自己的集合上使用一个商定好的密码学哈希函数计算哈希值，然后 R 发送给 S 自己的哈希值，S 计算交集。尽管这一协议是十分有效的，当输入域很小的时候存在着一方使用暴力破解攻击的可能（一方计算域上所有元素的哈希值，然后进行交集运算，从而得到另一方的所有输入）。但是当输入集合的取值范围很大时，比较哈希值的协议也可以被证明是安全的。

3.2 基于公钥加密体系的 PSI

在 [43] 中，Meadow 等人构建了一个使用 Diffie-Hellmann 密钥协商的 PSI 协议（相关的思想在 [58, 31] 中已经有所体现）。他们的协议可以看作是 Diffie-Hellman 密钥协商协议的简单应用，主要的用途是隐私保护场景下选择偏

好的匹配，也就是说，两方可以在保护各自输入私密性的前提下验证各自的输入是有某些程度上的匹配。

Freedman 等人在 [24] 中提出了一种在标准模型下（在基于 DH 的协议的分析中需要使用到随机预言假设），针对半诚实或是恶意的敌手安全的 PSI 协议。他们的协议是构建在多项式插值上的。在 [22] 中，Freedman 等人使用了基于模拟器的证明方法证明了这一协议在恶意敌手存在的情况下仍是安全的，并且测试了协议的运行效率。一种使用不经意伪随机函数（Oblivious Pseudo-Random Function）的方法在 [23] 中被提出。一种使用多项式插值和求导来计算多个集合之间交集的协议在 [36] 中被提出。

另一种使用盲签名（基于 RSA 公钥体系），并且计算和通信复杂度随着集合大小线性增长的 PSI 协议在 [13] 中被提出。在 [14] 中，作者给出了这一协议的实现与性能测试。在 [16] 中，作者提出了一系列使用布隆过滤器（Bloom Filter），实现了计算交集、计算交集基数和在交集上可认证计算的 PSI 协议。以上的协议中，公钥加解密操作的次数与集合大小成线性关系。因此虽然通信复杂度是最小的，但是计算开销远不及后面叙述的基于不经意传输扩展协议的 PSI。

使用公钥加密体系的 PSI 有另一个优点，就是在双方集合大小相差很大的情况下，花销很大的公钥加密操作可以集中在一方进行。结合这种方案通信复杂度低的优点，Chen 等人在 [10] 和 [9] 中提出了使用基于 RLWE 的同态加密构造的 PSI 协议。这两种协议在随机预言假设和半诚实模型下被证明是安全的。这两篇文章中的成果是目前集合大小非对称情况下（比如移动端用户寻找联系人的应用场景下）性能最好的 PSI 协议。

3.3 基于电路的 PSI

如第2.5小节所述，通用的安全多方计算方法在近年来得到了很大的优化。由于 PSI 协议解决的就是一类安全多方计算的问题，而且使用通用框架来计算交集有着容易扩展的特性。这是因为使用通用的安全多方计算协议，计算交集的函数是通过电路实现的，而将计算交集的电路的输出连到计算其他电路的输入就可以用来计算集合大小、集合中元素的求和等函数，而且全部过程都满足能够保护输入的隐私性。这种易于扩展的性质是其他基于公钥加密或是基于不经意传输扩展协议的 PSI 所没有的，因此这也是一种有实际应用价值的 PSI 方案。

Huang 等人在 [29] 中提出了几种可以用来实现 PSI 的布尔电路，并在 [30] 中通过混乱电路框架测试了这些方案。特别地，Huang 等人的结果表明他们使用 Java 的实现可以在在安全参数上的扩展性很好，并且在大安全参数的情况下有着比 [13] 中盲签名方案更好的性能。Pinkas 等人在 [52] 中提出了通过哈希表的方法对 Huang 等人的方案做出的优化，并且比较了使用不经意伪随机函数与之前提出的布尔电路的性能比较。

Ciampi 等人在 [12] 中给出了基于两方安全计算的 PSI 协议，这一协议拥有比 Pinkas 等人给出的优化结果（基于通用框架的协议）更好的性能，是目前使用通用安全多方计算框架下性能最好的协议。

3.4 基于不经意传输扩展协议的 PSI

如第2.5小节所述，不经意传输协议扩展的出现使得 OT 协议的性能有了极大的提升。借助这一工具，许多基于 OT 协议的 PSI 达到了与不安全的哈希协议复杂度在同一量级的性能。

在 [19] 中，Dong 等人提出了使用布隆过滤器和 OT 扩展协议的 PSI 协议。构造出的协议拥有可以在亿级规模的集合上操作，并且可以证明是在半诚实模型和恶意模型下是安全的。不过在 [40, 54] 中，Rindal 等人提出使用布隆过滤器的 PSI 协议在恶意模型下存在攻击的可能，并在 [53] 给出了如何改进现有的使用布隆过滤器的协议，从而给出了第一个在恶意模型下安全的 PSI 协议。改进后的协议可以在约 $200s$ 的时间内计算两个大小为一百万的集合的交集。

在 [54] 中 Rindal 等人给出了之前 Pinkas 等人提出的 PSI 协议的改进。改进后的协议达到了恶意模型下的安全性。在 [40] 中，Lambæk 给出了 [48] 中 PSI 协议（基于 OT 协议）的改进，改进后的协议可以在一方是半诚实的，一方是恶意的情况下证明安全性。在 [55] 中 Rindal 等人给出了使用基于 Dual Execution 思想对基于 OT 扩展的 PSI 协议的安全性加固，加固后的协议具有恶意模型下的安全性。在经过哈希表、Phasing 等方法优化后，协议达到了比之前最好的恶意模型安全 PSI 协议 [54] 更好的性能（在集合大小对等的情况下）。在第4中给出了 Rindal 等人给出的测试环境说明以及测试结果的展现。

以上是针对恶意模型的协议改进，在半诚实模型下，由于敌手的能力仅限于在正常的协议交互过程中尽最大程度地获取另一方的输入，这种类型的协议的效率可以达到比之前恶意模型下安全的协议更好的效果。在 [38] 中，Kolesnikov 等人给出了 [50] 中 PSI 协议的性能改进，主要的改进方法是使用 OT 扩展协议来实现不经意伪随机函数的构造，并且将 [50] 中使用到的纠错编码方法改为伪随机编码（因为并不需要配套的解码过程）。性能改进的主要效果是协议的通信复杂度不再依赖于集合元素的长度 σ。在 [52] 中 Pinkas 等人继承了 [38] 中体现的使用不经意伪随机函数的思想，但是保留了纠错编码的构造，因此在集合元素长度 σ 较小的情况下能达到比使用伪随机编码更好的性能，但是在 σ 增大的情况下，协议通信复杂度也会随之增大。Pinkas 和 Kolesnikov 等人的结果是在半诚实模型下，集合元素

大小对等的场景下性能最好的 PSI 协议，与不安全的哈希协议的开销在同一量级或是只差一个数量级。(Kolesnikov 等人再 [38] 中的实验结果表明如此，但是由于缺乏在同一实验环境下的测试数据，在第4节中并没有给出相应的比较结果。)

3.5 基于全同态加密的 PSI

全同态加密是一种功能强大的加密原语，允许运算电路直接在密文上计算，而不必首先解密数据。乍一看，使用完全同态加密似乎很容易实现低通信成本的 PSI 协议。拥有较小集合大小的一方将数据加密后发送给另一方，另一方利用全同态在交叉电路上计算，然后将结果发回给第一方进行解密。总共的通信量仅为

$$2\times\text{密文扩展}\times\text{较小集的大小}.$$

然而，仅仅单纯实现上述想法上述想法非常低效，因为对于所有已知的完全同态加密方案，计算成本不仅随着输入大小（两个集合大小的总和）的增长而增长，也随着电路深度的增长而迅速增长。因此，我们的主要挑战是提出各种优化措施，使解决方案切实可行，甚至在许多情况下，能比最先进的协议还要快。在论文 [11] 中构造了一种高效的基于全同态的 PSI 协议，该协议拥有很小的通信量方面的代价。该协议尤其适用于寻找联系人的应用场景。

4 性能分析与比较

在这一节中我会给出不同 PSI 方案在实现中可能会出现的问题，比如内存的消耗和并行计算的可能。除此之外，我还会给出相应的性能分析与比较，包括在第3节中列举出的目前在不同场景下性能最好的 PSI 协议。Pinkas 等人在 [52] 中在统一的测试环境下，对大部分 PSI 协议的性能给出了具有参考性的测试结果，我会在这里将结果呈现出来。对于 [52] 中没有给出的协议，我会加入额外的比较。

4.1 实现 PSI 协议

由于现实情况下 PSI 协议通常要在很大规模的集合上操作，因此分析协议实现中可能遇到如内存紧张的情况。Pinkas 等人在 [52] 中分析了在实现协议的过程中可能遇到的问题以及一些工程上的优化方法。我在这里将给出阐述。

4.1.1 内存占用情况的分析

在对协议复杂度分析时，计算和通信复杂度通常是衡量协议效率的指标。此时认为能够使用的随机存取空间是够用的。但是在实现协议的过程中，内存是受限的。因此如果一个协议运行过程中涉及到对基数很大的集合进行操作，并且对于内存的需求超出了内存内提供的空间，协议的性能会大幅下降。因此对协议的内存使用情况的研究对于实现协议十分重要。

在 [52] 中，Pinkas 等人观察到在大集合上进行密码学原语的操作会耗费大量的内存。出于这种原因，很多 PSI 协议的实现版本如果不经优化的话会很快地消耗掉所有的内存。将数据存储在硬盘上是一种可行的方法，但是如果需要在数据上做随机访问时这种方法会带来很大的性能损失。

下面我会分析不同种类的 PSI 协议对内存的消耗情况。

安全性有限的和基于公钥的协议 基于 Hash 函数的非安全协议和使用公钥的 PSI 协议不会占用大量的内存。这是因为它们每次只对一个元素处理，并且很容易做到流水线操作。在标准的个人电脑上也可以在百万级的集合上运行这类协议。

基于电路的 PSI 基于电路的 PSI 会占用大量的内存空间，而且姚氏混乱电路相比 GMW 的方案，有着更大的内存占用情况，因为每条线路上都要存储 κ 个比特，而不是一个比特。电路方案造成大量内存占用的主要原因是计算需要的电路要完整地存储在内存中。为了减少内存的占用，在 [52] 中 Pinkas 等人给出的基于电路的 PSI 实现中，为了减少内存占用情况，求值过并且不会再使用到的电路都会被直接删除。而且 Pinkas 等人提出将一个电路重复使用的方法，这种优化方法对于 PWC 和 OPRF 类型的电路尤其有效，因为在这些情况下所有的数据都是应用于同一电路上的。

基于不经意传输扩展协议的 PSI 在 [19, 50] 中基于布隆过滤器的 PSI 协议需要将布隆过滤器全部存储在内存中才能进行取交集操作。具体来说，当输入集合有一百万个元素的时候，这些方案要求存储至少 875 MB 的数据。除此之外，在布隆过滤器上还需要进行随机访问操作，这样一来如果将布隆过滤器存储在硬盘上，性能会有很大的损失。

基于 OT 协议的 PSI 的主要内存需求出现在其中使用的哈希表中，尤其是使用了布谷鸟哈希（Cuckoo Hashing）之后，在构建哈希表的时候需要对表进行随机的访问。与布隆过滤器的情况类似，如果将一张很大的表存储在硬盘上会造成很大的性能损失。但是哈希表的实际表现要优于布隆过滤器。具体来说，对于存储一百万个元素的情况，布谷鸟哈希表需要 12 MB 的空间，因此要远优于布隆过滤器的方案。

4.1.2 并行化

当 PSI 协议的性能瓶颈出现在计算过程的时候，通过将并行计算来加速计算过程就成为了一种切实可行的加速手段。注意由于对称加密可以通过 AES-NI 指令集快速完成，在这种情况下并行化很可能将性能瓶颈从计算过程转化为通信过程。下面将讨论不同种类协议的并行化可能。

安全性有限的和基于公钥的协议 这两类协议都可以使用并行计算，因为元素的处理是相互独立的。这些协议中主要的性能瓶颈是在协议最后，哈希值上的交集操作。

基于电路的 PSI 基于电路的 PSI 会根据底层安全多方计算框架的不同，而有着不同的并行化手段。GMW 协议使用 OT 扩展来预计算一种称为 “multiplication triple” 的结构。预计算步骤占计算开销的主要部分，并且是可以并行化处理的。但是在 GMW 中，电路的计算需要两方顺序地交互，交互次数与电路的深度呈线性关系，是不可以并行化处理的。

另一方面，姚氏混乱电路有着常数的交互轮次。协议并行化的能力依赖于底层的电路结构。如果电路是可以被划分为许多独立的子电路，比如 PWC 和 OPRF 电路，那么就可以有效地进行并行处理。如果电路中所有的门都是有关联的，比如 SCS 电路，做到并行化就需要和电路结构相关的方法来进行，比如在 [8] 中提出的自动并行化电路编译器。

基于不经意传输扩展协议的 PSI 所有基于 OT 协议的 PSI 都隐藏着并行化的可能，因为底层的 OT 扩展协议是可以并行化处理的。不同协议的并行化能力的主要差距体现在将集合中元素映射到相应的数据结构的方法（比如布隆过滤器和哈希表）。在 [19] 中使用布隆过滤器的方案中，X 需要首先提前生成布隆过滤器，这一步骤是不能很好地并行化的。在 [50] 中，这一点得到了改善，因为在这一协议中布隆过滤器是通过 OT 扩展协议的输出产生的，因而可以并行化。在 [52] 提出的基于 OT 的 PSI 协议中，并行化主要的瓶颈存在于布谷鸟哈希的步骤。但是由于在这一协议中布谷鸟哈希是作为预处理步骤进行的，无需和另一方交互，因此在某些应用场景下这一开销可以均摊到多次协议的实例上。

基于全同态加密的 PSI 全同态加密技术使得对于明文的数学操作可以直接在密文上进行而不需要先将密文解密。早期的全同态加密十分低效，而它的性能在近几年才有所提高。使用全同态加密技术的 PSI 协议，通过拥有较小集合的一方将自己的集合加密以后发送给另一方，而另一方负责基于密文求两个集合的交集，然后将结果交给对方解密。使用全同态加密实现 PSI 可以达到相对较小的交互复杂度，但它的计算复杂度通常非常高，导致协议效率低下。所以，在不牺牲太多交互复杂度的情况下降低计算复杂度是基于全同态加密的 PSI 协议主要面临的挑战。

4.2 PSI 协议的性能分析

不同 PSI 协议的计算复杂度和通信复杂度在表2中有所示。表中的计算复杂度是通过非对称或是对称密码原语的使用次数衡量的，通信复杂度是通过在信道上传输的比特数衡量的。这里的假设是每完成一次 OT 协议花费 3 次对称密码操作（对于使用布隆过滤器的花费 2.5 次对称密码操作）。计算姚氏电路中的与门使用 4 次对称密码操作，计算 GMW 电路中的与门使用 6 次对称加密操作。

在同一类别中的 PSI 方案大多数拥有类似的复杂度。朴素哈希方法与服务器辅助的方法需要对每一个元素执行一次对称加密操作（哈希），基于公钥的协议需要对每一个元素执行两次公钥操作，并且需要发送两个密文和一个哈希值。基于电路的方法的计算复杂度与电路中与门的数量成正比，在基于布隆过滤器的协议中，计算复杂度与布隆过滤器的大小成正比。在基于 OT 的协议中，基于布隆过滤器的协议 [19]，通信复杂度是与安全参数 κ 的平方成正比的，但是在 [52] 中的协议，通信复杂度是与 κ 呈线性关系。

分类	协议	计算复杂度（对称或非对称密码原语的次数）	通信复杂度（比特）
安全性受限的	朴素哈希	m sym	$N_X v$
	服务器辅助的 [34]	m sym	$t + \|X \cap Y\|$
基于公钥体系的	基于有限域上 DH 的 [43]	$2t$ pk	$t\rho + N_X v$
	基于椭圆曲线上 DH 的 [43]	$2t$ pk	$t\phi + N_X v$
	基于 RSA 的 [13]	$2t$ pk	$t\rho + N_X v$
基于电路的	姚氏电路使用 SCS	$12m\sigma \log m + 3m\sigma$ sym	$6m\kappa\sigma \log m + 2m\kappa\sigma$
	GMW 使用 SCS	$18m\sigma \log m$ sym	$6m(\kappa+2)\sigma \log m$
	姚氏电路使用 PWC	$\sigma(4\epsilon N_Y max_b + 4sN_X + 3\epsilon N_Y)$ sym	$2\epsilon N_Y \kappa max_b \sigma + 3sN_X\kappa\sigma + 2\epsilon N_Y\sigma$
	GMW 使用 PWC	$6\sigma(\epsilon N_Y max_b + sN_X)$ sym	$2(2+\kappa)\sigma(\epsilon N_Y max_b + sN_X)$
	姚氏电路使用 OPRF	$21760N_Y + 3\sigma N_Y$ sym	$10880N_Y\kappa + 2N_Y\kappa\sigma + N_X v$
	GMW 使用 OPRF	$32640N_Y$ sym	$10880N_Y(\kappa+2) + N_X v$
	Ciampi 等人的方案	$m(4\sigma \log m + 3\sigma)$ sym	$m(2\sigma m\kappa + m\kappa)$
基于 OT 协议的	使用布隆过滤器 [19]	$3.6m\kappa$ sym	$1.44m\kappa(\kappa+\lambda)$
	使用哈希表 [52]	$3\epsilon N_Y + (k+s)N_X$ sym	$512\epsilon N_Y + (k+s)N_X v$
	使用哈希表 [38]	$(s+3)m + (\epsilon m + s)$ sym	$4(\epsilon m + s)k + (s+3)mv$
	使用哈希表 [55]*	-	$6\kappa n + \beta n \log n$
基于 FHE 的 [11]	-	-	$1.5C\sigma N_Y \log_2 N_x$

表 2: 不同 PSI 协议的复杂度比较，其中 sym 与 pk 分别表示对称与非对称操作的统计，$t = N_X + N_Y, m = \max(N_X, N_Y)$，$\beta \approx \lambda + 2\log n - 1$，$\epsilon, k, s, max_b$ 是哈希函数用到的参数，v 是在 OT 扩展协议中，使用的哈希函数的输出长度，C 是一个常数，表示 [11] 的同态操作产生的密文扩展。标有 * 的是在恶意模型下安全的协议。

4.3 性能测试结果

在这一节中我会给出在局域网和广域网场景下不同 PSI 协议的性能。衡量性能的指标包括总体的运行时间以及在信道上传输的数据量。这里的测试数据来源于 Pinkas 等人在 [52] 中以及 Chen 等人在 [11] 中给出的性能比较。由于不同作者进行的实验环境与软件实现都可能存在差异，因而无法做到并列地比较。但是好在每一组数据中都考虑了足够多的协议种类，可以通过每一组数据的观察来比较相应类别的 PSI 协议的特点与适用场景。

测试环境说明 Pinkas 等人在局域网上的测试是在两台通过千兆以太网连接的 PC 机（Intel Haswell i7-4770K CPU with 3.5 GHz and 16 GB RAM）上进行的，而广域网上的测试是在两台 Amazon EC2 m3.medium 实例（Intel Xeon E5-2670 CPU with 2.6 GHz and 3.75 GB RAM）上运行的，其中一台位于美国西海岸的北弗吉尼亚州，一台在欧洲的法兰克福。两者之间通信网的平均带宽为 $98MBit/s$，平均的往返延迟为 $94ms$。

在局域网和广域网的条件下，Pinkas 等人分别运行了测试。在测试中 Sender 和 Receiver 拥有相同数目的输入元素个数，取值范围是

$$N_X, N_Y \in \{2^8, 2^{12}, 2^{16}, 2^{20}, 2^{24}\}$$

在测试中双方都没有进行预计算。对于基于电路的 SCS 和 PWC 方案，它们的计算复杂度取决于输入元素的长度 σ。在测试中 Pinkas 等人固定 $\sigma = 32$ 并给出了 IPv4 地址的 PSI 的应用场景。对于服务器辅助的 PSI，Pinkas 等人的实现方法是在一台机器上运行服务器程序，并在另外一台机器上运行两个希望进行 PSI 的客户端程序。

对于协议的实现，基于 RSA 的协议 [59] 和基于布隆过滤器 [19] 的协议都是用了原作者给出的实现方案，但是在基于 RSA 协议的最后一步 Pinkas 等人使用了哈希表进行优化（原来的协议是单个元素逐个比较，复杂度与输入元素的个数的平方成正比），在基于布隆过滤器的协议中，使用了 Asharov 等人给出的 OT 扩展协议 [1] 来实现原协议中的 OT 协议。他们使用了使用 C++ 实现的，当时最好的姚氏电路和 GMW 协议的 ABY 框架 [17] 来实现基于电路的协议。对于姚氏电路，Pinkas 等人使用了对于电路尺寸优化的 SCS 电路（比较部分的电路的大小和深度为 σ）；对于 GMW 协议，他们使用了对深度优化的电路（比较部分的电路大小为 3σ，深度为 $\log_2 \sigma$）。这两种优化思路来自 [56]。他们使用了 AES 来作为服务器辅助 PSI 中 PRF 函数 [34] 和二选一 OT 扩展协议中的 CRF 函数，使用 SHA-256 作为 N 选一 OT 扩展协议中 CRF 函数和随机预言模型的实现。

对于协议中使用到有限域上的密码算法，Pinkas 等人使用了 GMP 库（v. 5.1.2）；对于椭圆曲线上的密码算法，使用了 Miracl 库（v. 5.6.1）；对于对称密码算法使用了 OpenSSL 库（v 1.0.1e）；对于 OT 扩展协议，使用了 [1] 中 Asharov 等人给出的实现。在有限域上的密码算法都是在阶为 q 的子群上进行的，其中 q 的长度为 2κ bit。

表3展现了不同协议的运行时间，表4展现了不同协议的通信量。其中表3中的运行时间是从程序开始运行到接收方输出交集结果的时间，所有的运行时间都是十次运行的算术平均值。表3与表4中的数据来自于 [52] 中的测试结果，其中对于 [38] 和 [55] 中的协议，Pinkas 等人并没有给出测试与比较。但是这两篇论文中给出的性能数据所使用的测试环境与以上的测试环境不同，并没有可比性，因此我们在表3与表4中并没有给出这两种协议的性能数据。下面我们按类别分析比较的结果。

Chen 等人在 [10] 中使用的测试环境如下：基准测试机有两个 18 核 Intel Xeon CPU E5-2699 v3 @ 2.3GHz 和 256GB RAM，并使用 Linux 流量控制器命令模拟网络延迟和带宽。具体来说，LAN 设置为双方通过本地主机连接，具有 10Gbps 吞吐量和 0.2ms 的往返时间（RTT）；WAN 设置方面分别考虑了 100Mbps，10Mbps 和 1Mbps 的带宽，每种设置的 RTT 均为 80ms。所有数据为 10 项试验的平均值。

表5是论文 [10] 中分别基于 FHE、电路和不经意传输的 PSI 协议运行性能比较。

Rindal 等人在 [55] 中比较了他们在恶意模型下安全的协议与其他恶意模型下安全的协议的性能，以及与基于 OT 扩展的半诚实模型协议 [38] 的性能。他们的测试环境如下：基准测试机有两个多核的 Xeon 处理器，256GB 的 RAM。测试中 LAN 和 WAN 网络都是使用本机 loopback 结合 Linux 流量控制器命令来模拟网络延迟和带宽的。其中 LAN 设置具有 10Gbps 的吞吐量和小于 1ms 的往返时间；而 WAN 设置使用的是 40Mbps 的吞吐量和 80ms 的往返时间。所有的测试都使用了 $\kappa = 128$，$\lambda = 40$ 的安全参数，而且运行时间是十次运行的算术平均。其中任何可能使用硬件优化的地方都使用了 AES-NI 指令进行优化。[3]

在实验中，Rindal 等人使用 [45] 作为不经意编码功能的实现，而且使用的编码方式为 BCH-(511, 76, 171)。这一模块的输入长度为 76 bits。为了支持任意长度的输入元素，Rindal 等人使用了 Pinkda 等人在 [52] 中给出的变换方法，并配合 Phasing（在哈希表插入时进行置换从而减少表元素的长度）使用，最终的元素长度为 128 bits。在这种情况下，集合大小 n 和统计参数 λ 的关系为

$$\lambda + \log n \leq 76$$

对于 RO 和 PRF/PRG，Rindal 等人分别使用了 SHA-1 和 AES 作为实现（使用 PRG 的计数器模式）。

表6给出了 Rindal 等人在 [55] 中提出的两种恶意模型下安全的协议、Rindal 等人在 [54] 中给出的恶意模型下安全的协议以及 Kolesnikov 等人在 [38] 中给出的一个半诚实模型下安全的协议的比较。

安全性有限的协议 朴素哈希是最高效的协议，比服务器辅助的 PSI 快将近两倍。但是这一类中的协议需要做着更强的安全性假设或是根本就是不安全的。

基于公钥体系的 PSI 基于有限域上 DH 的 PSI 协议 [43] 比基于 RSA 体系的 PSI[13] 性能更好，而基于椭圆曲线上 DH 的 PSI 协议比基于有限域 DH 的 PSI 协议的性能又有近两倍的提升。基于椭圆曲线上 DH 的 PSI 协议的主要优势是它拥有最低的通信复杂度，而且协议比较简洁，易于实现。

基于电路的 PSI 在表3中罗列了通过使用姚氏电路和 GMW 实现的 PWC 电路，SCS 电路和 OPRF 电路的性能。比较结果显示使用 GMW 实现的协议相比姚氏电路，拥有约两倍的性能提升。PWC 电路在集合基数增大的情况下有比 SCS 和 OPRF 电路更好的性能，例如在集合大小为 2^{16} 的时候 PWC 电路的运行时间至多是其他电路的三分之一。

基于 OT 协议的 PSI 在使用 OT 的 PSI 协议中，当在小集合上操作时，使用哈希表的 PSI 协议比使用布隆过滤器的协议有着更长的运行时间。这是因为使用哈希表的协议中，基础 OT 协议的运行次数为后者的三倍。但是当集合大小增大时，基础 OT 协议的开销被分摊到更多的扩展 OT 实例中，基于哈希表的 PSI 协议的优势也就变得明显了。

Rindal 等人给出的比较表明在超过一百万的集合上进行 PSI 操作时，恶意模型下安全的协议要比半诚实模型下安全的协议有至少一个数量级的性能损失。但是在一些对安全性要求非常严格的应用场景下，这些协议可以提供更加可靠的安全保证。

基于 FHE 的 PSI 与基于电路的方案比较来看，基于 FHE 的 PSI 协议主要优势在于非对称的场景，不仅对接收方的计算能力有更小的要求，而且在多线程场景下速度有大幅度提升。而在与基于 OT 的比较来看，基于 FHE 的 PSI 协议的通信量远远小于基于 OT 的 PSI 协议，所以在 WAN 的网络设置下，基于 FHE 的 PSI 协议将比基于 OT 的协议快 57 倍以上。

[3] Rindal 等人的代码可以在`https://www.github.com/osu-crypto/libPSI`访问到。

类型	网络场景	LAN					WAN			
	集合大小	2^8	2^{12}	2^{16}	2^{20}	2^{24}	2^8	2^{12}	2^{16}	2^{20}
有限的安全性	朴素哈希	1	3	38	665	12368	51	119	886	7277
	服务器辅助的 [34]	1	5	78	1250	20053	124	248	1987	15578
基于公钥体系	基于有限域上 DH 的 [43]	386	5846	88790	1418772	22681907	3577	56786	880075	11557061
	基于椭圆曲线上 DH 的 [43]	231	3238	51380	818318	13065904	1949	28686	466606	5007681
	基于 RSA[13]	779	12546	203036	3193920	50713668	10508	166453	1356757	21094586
基于电路的	姚氏电路使用 SCS	320	3593	74548	-	-	2763	20826	518136	-
	GMW 使用 SCS	361	1954	40872	-	-	5929	14415	187750	-
	姚氏电路使用 PWC	304	1647	19080	-	-	3115	12189	121198	-
	GMW 使用 PWC	325	905	7085	83889	-	2086	5881	43353	337851
	姚氏电路使用 OPRF	968	12518	-	-	-	6001	65156	-	-
	GMW 使用 OPRF	690	6672	101231	-	-	6939	27660	386243	-
基于 OT 协议的	基于布隆过滤器 [19]	105	448	4179	75218	-	1248	5424	31581	345484
	基于哈希表 [52]	309	339	658	5680	83739	2211	2809	7857	56738

表 3: 在局域网和广域网场景下的 PSI 协议的运行时间，其中运行时间以毫秒计，“-” 表示在测试的时候遇到内存不足的情况，实验无法继续

类型	网络场景	WAN 或 LAN				
	集合大小	2^8	2^{12}	2^{16}	2^{20}	2^{24}
有限的安全性	朴素哈希	0.002	0.031	0.600	10.000	176.000
	服务器辅助的 [34]	0.003	0.063	1.133	20.125	354.000
基于公钥体系	基于有限域上 DH 的 [43]	0.195	3.125	50.000	800.000	12800.000
	基于椭圆曲线上 DH 的 [43]	0.020	0.280	4.560	74.000	1200.000
	基于 RSA[13]	0.195	3.125	50.000	800.000	12800.000
基于电路的	姚氏电路使用 SCS	7.522	168.590	3484.751	-	-
	GMW 使用 SCS	7.319	162.851	3348.011	-	-
	姚氏电路使用 PWC	6.923	93.371	1220.194	-	-
	GMW 使用 PWC	4.320	57.864	749.421	9169.917	-
	姚氏电路使用 OPRF	44.033	704.210	-	-	-
	GMW 使用 OPRF	43.193	690.890	11054.050	-	-
基于 OT 协议的	基于布隆过滤器 [19]	1.037	17.314	288.560	4801.639	-
	基于哈希表 [52]	0.055	0.424	6.500	107.000	1757.000

表 4: 测试中 PSI 协议的通信量，其中通信量以 MB 计，“-” 表示在测试的时候遇到内存不足的情况，实验无法继续

参数		协议	通信量	运行时间 (s)							
N_x	N_y		大小 (MB)	10 Gbps		100 Mbps		10 Mbps		1 Mbps	
				$T=1$	4	1	4	1	4	1	4
2^{24}	11041	基于 FHE 的 [11]	23.2, †21.1	115.4	40.3	117.8	42.7	134.4	59.3	†290.8	†215.1
		基于电路的 [51]	480.9	40.5	23.3	88.0	66.4	449.5	427.5	4084.8	4067.2
		基于 OT 协议的 [39]	975.0	70.8	–	188.7	–	1269.1	–	12156.7	–
	5535	基于 FHE 的	20.1, †12.5, ‡11.0	105.2	34.8	107.2	36.7	†120.3	†45.8	†211.1	‡132.7
		基于电路的	480.4	40.1	23.1	87.9	65.5	449.2	427.3	4080.6	4064.3
		基于 OT 协议的	962.1	70.4	–	188.3	–	1263.5	–	12153.2	–
2^{20}	11041	基于 FHE 的	11.5	12.8	5.7	14.0	6.9	22.2	15.1	105.4	98.3
		基于电路的	30.9	3.3	2.1	7.0	5.6	29.8	28.3	263.7	262.1
		基于 OT 协议的	58.5	4.5	–	11.6	–	79.4	–	688.1	–
	5535	基于 FHE 的	5.6	8.6	3.3	9.2	3.9	13.3	8.0	53.6	48.3
		基于电路的	30.4	3.1	2.0	6.8	5.0	29.0	27.9	260.0	259.6
		基于 OT 协议的	57.3	4.4	–	11.5	–	79.3	–	686.0	–
2^{16}	11041	基于 FHE 的	4.1, †4.4	3.0	†1.7	3.4	†2.1	6.4	†5.3	36.0	35.0
		基于电路的	2.6	0.7	0.6	1.5	1.4	3.3	3.1	21.6	22.1
		基于 OT 协议的	4.5	0.4	–	1.4	–	5.6	–	48.2	–
	5535	基于 FHE 的	2.6	1.8	0.9	2.0	1.2	3.9	3.1	22.5	21.7
		基于电路的	2.1	0.7	0.6	1.4	1.3	2.9	2.8	19.8	21.3
		基于 OT 协议的	3.7	0.4	–	1.2	–	5.4	–	46.7	–

表 5: 基于 FHE 的 PSI 分别与基于电路和 OT 协议的 PSI 在通信量 (MB) 以及运行时间 (s) 方面的比较，其中 T (线程) $\in \{1, 4\}$; λ (安全参数) $= 40$, σ(集合元素长度) $= 32$, h (Cuckoo Hashing 参数) $= 3$. $10Gbps$ 的网络并且 $RTT = 0.2ms$.

网络场景	协议	集合大小				
		2^8	2^{12}	2^{16}	2^{20}	2^{24}
LAN	半诚实模型下的 [38]	0.19	0.21	0.4	3.8	59
	使用布隆过滤器的 [54]	0.21	0.8	9.6	148	-
	使用 Encode-Commit 方法 [55]	0.13	0.19	0.94	12.6	239
	使用 Dual-Execution 方法 [55]	0.13	0.23	1.3	18	296
WAN	半诚实模型下的 [38]	0.56	0.59	1.3	7.5	107
	使用布隆过滤器的 [54]	0.97	5.3	69	1080	-
	使用 Encode-Commit 方法 [55]	0.67	1.5	16	255	3208
	使用 Dual-Execution 方法 [55]	0.9	1.2	6.3	106	2647

表 6: Rindal 等人在 [55] 中给出的实验结果，其中运行时间为总的运行时间，以秒计。[38] 中的协议是在半诚实模型下安全的，而表中其他协议是在恶意模型下安全的，前者代表半诚实模型下优化最好的结果。标 “-” 的表示运行时间超过 24h 或者发生了内存溢出

5 基于 MesaTEE 的 PSI

5.1 MesaTEE 简介

MesaTEE 是全球首个通用安全计算平台（Universal Secure Computing，简称 USC）。它为对安全和隐私有强诉求的场景提供了下一代通用安全计算能力，使得敏感数据即便在企业外环境和离岸场景下也能安全受控的流通和处理，而不会被泄漏或者滥用。这在全球关注隐私的今天格外重要，使得很多大数据业务成为可能。同时，由于

USC/MesaTEE 天生的弱中心架构，也使得它和区块链形成完美的互补，填补了区块链急缺的高性能隐私数据处理能力。

MesaTEE 综合采用三项核心安全技术，包括百度安全实验室提出的混合内存安全技术（Hybrid Memory Safety 与 Non-bypassable Security Paradigm），机密计算技术（Confidential Computing，如 Intel SGX），以及可信计算技术（如 TPM），构建了完整的 FaaS 通用计算框架，提供了严密而实用的隐私和安全保障能力。与传统基于密码学的多方安全计算或全同态加密技术相比，USC/MesaTEE 性能一般会快百倍以上，而且编程模式与传统编程一致，适合普通程序员快速上手进行应用。后继版本也会支持 MesaPy（百度安全实验室推出的内存安全的 Python）等安全语言，进一步降低开发门槛。如图1所示，MesaTEE 通过提供可信且安全的隔离执行计算环境，重新定义了未来的

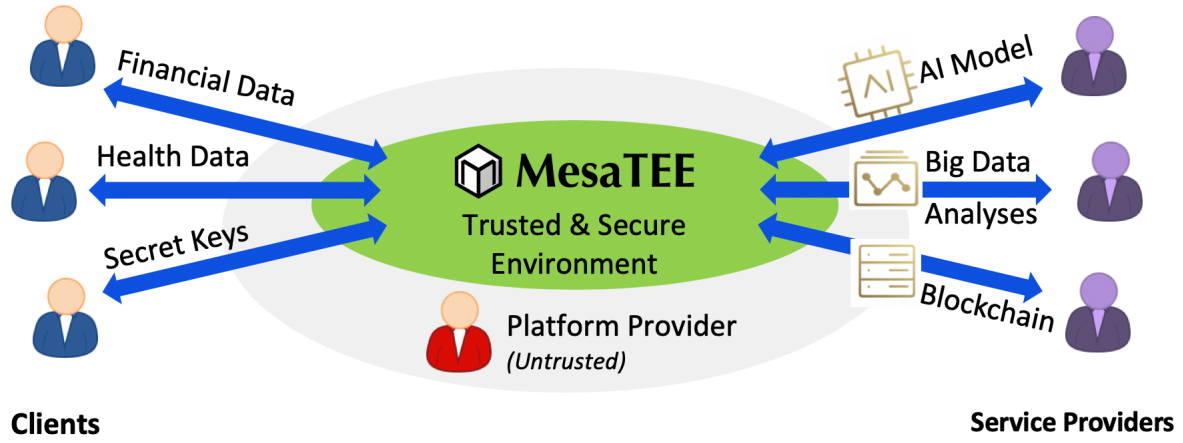

图 1: MesaTEE Overview.

大数据商业模式。即使客户端和服务/平台提供商不完全相互信任，也可以有效地保护数据或模型的机密性和完整性。同时，MesaTEE 大大简化了可信计算基础（TCB）、信任边界、和信任模型复杂度，让整个软件栈的审计和验证变得实际可操作。

5.2 基于 MesaTEE 的 PSI 协议

目前 MesaTEE PSI 是以 Intel SGX 为信任根的基础上搭建起来的。Intel SGX 提供了根植于 CPU 的硬件可信和高强度的隔离运行环境（enclave）。即使攻击者控制了操作系统和其他特权级软件，也无法直接访问 enclave（即无法修改也无法读取信息）。Intel SGX 还提供远程认证服务，即 enlave 中的程序向其第三方证明其可信性。基于以上特性，我们可以基于 SGX 建立一个可信的可验证的 PSI server。这样，其他 PSI 的参与方都可以通过远程认证来验证 PSI server 是否处于可信状态。

图2给出了一个 MesaTEE PSI 协议的流程图。图中该协议虽然以两个参与者为例子，但该协议兼容并且很容易扩展到三个或者多个参与者的情况。

- **第一步**：参与 PSI 的各个参与方（两方或者多方）和 PSI 服务器通讯，服务器为各个参与方产生随机 salt。
  - 参与方对服务器进行远程认证，确保服务器处于可信状态。
  - 服务器通过硬件指令生成随机 salt，比如通过 rdrand 指令。
- **第二步**：服务器通知参与方，让参与方通过生成的 salt 对数据进行散列。
  - 服务器把 salt 发回各个参与方。
  - 各个参与方对自己的数据和 salt 进行散列，使得攻击者无法通过 hash 结果反推原始数据。
  - 排序散列之后的数据。因为 salt 的使用，排序结果仍然打破 hash 和原始数据对应的 order 关系，使得侧信道攻击更加困难。
- **第三步**：参与方向服务器传输处理好的数据并要求服务器进行 PSI 操作。
  - 参与方把排序后的随机盐 hash 数据加密传回服务器，并使用不同的传输密钥。
  - 服务器在接收各个参与方的数据后开始做 PSI。由于是排序的 hash 数据，所以可以做流式求交，并且可以分布式并行处理。

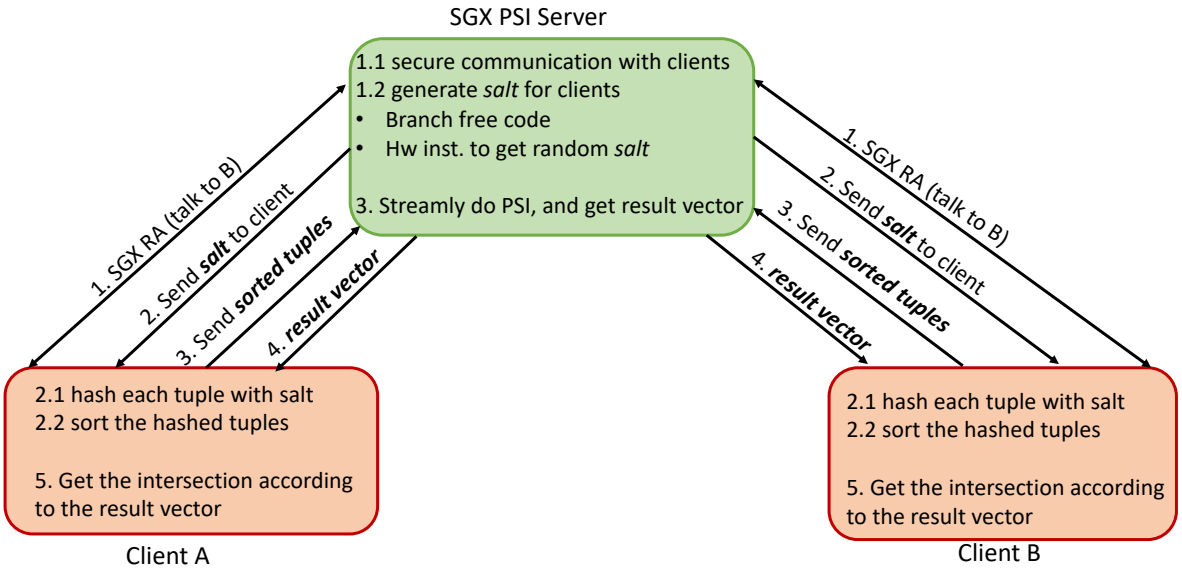

图 2: MesaTEE PSI 协议。该协议还具有抵抗 SGX 侧信道攻击的能力。

  - 服务器通过一个 bit vector 来表达对 PSI 结果。这样做的目的同样是缓解侧信道攻击。

- **第四步**：服务器传 vector 给各个参与方。每个参与方获得的 vector 是不同的。
- **第五步**：参与方通过自己的 vector 来获得最终的 PSI 结果。

该基于 SGX 的 PSI 协议不仅高效，而且具有很高的安全性：1）具有 SGX 的强隔离和可信等特性；2）缓解了潜在的 SGX 自身的侧信道攻击。使得该协议比单纯的 Intel SGX 具有更高的安全性。

5.3 MesaTEE PSI 的优势

传统的 PSI 是基于密码难题来构建可信根的。另外一种构建可信根的技术就是使用可信硬件，比如 Intel 的 SGX 就是使用 CPU 作为可信根（如图3)。使用 SGX 作为可信根的好处是：1）SGX 的可信根比以往其他的可信根（比如 TPM）都要更小更可信，2）SGX 提供一个安全可信的运行环境，从而保障代码和数据的机密性和完整度；3）SGX 可信运行环境可以提供近似原生态的运行速度，从而大大降低运行时期的性能开销。

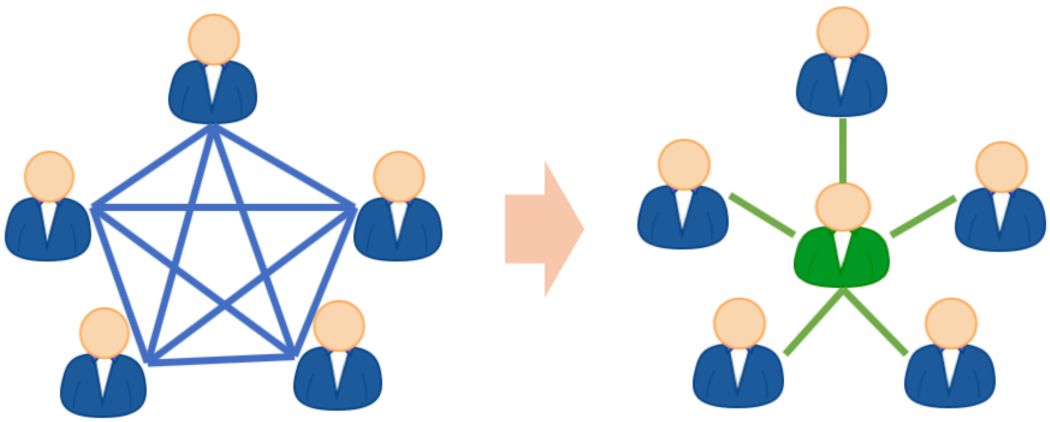

图 3: 传统 PSI（左图）和 MesaTEE PSI（右图）.

在此基础上，MesaTEE PSI 还进一步加强了系统的安全性：

- 保证里面运行软件的内存安全性，使得诸如 use after free、double free、buffer overflow 等内存安全问题不会发生。
- MesaTEE 采用了“不可绕过范式”（Non-bypassable Security Paradigm），约束所有控制流和数据流必须经过关键检查点，显著减轻了审计和访问控制的难度，极大缩小了攻击面，归约了访问控制策略的部署，也让以此为基础的安全形式化验证变得实际可行。
- 采样抵御侧信道攻击的编程模式，使得潜在的侧信道攻击变得异常困难甚至无法完成。

相比于基于 SGX 的 PSI，传统的 PSI 有着诸多的不足：

- 对于指定的协作运算，一旦预处理结束那些保密的信息只能被预先指定的参与方使用。缺少灵活的增加或减少参与方的能力。MesaTEE PSI 就没有这样的限制。
- 对于指定的运算模式，一旦预处理结束各方就只能做之前确定好的运算。缺少灵活的增加运算的能力。MesaTEE PSI 就没有这样的限制。
- 每一个参与方都需要互相交换信息，这样就造成了性能损失和无法避免的高延迟。MesaTEE PSI 只需要和中心可信节点通讯，有效的避免了这些问题。
- 传统的 semi-honest 的 PSI 不具有对运算结果的完整性保证。MesaTEE 的 PSI 具有天然高效的完整度保护。从这个角度出发，MesaTEE PSI 比这类传统 PSI 具有更高的安全性。如果要扩展传统 PSI 增加完整度保护（比如扩展到 malicious model 的情况），会使得整个协议的性能有数量级程度的下降。MesaTEE 仍可以匹配这类最强安全强度的传统 PSI。
- SGX PSI 可以自适应 2 方，3 方以及任意不同数量的参与者，传统 PSI 在这个自适应方面就明显不足。

在此之外，相比于传统的 PSI 方案，MesaTEE PSI 在性能方面也有着明显的优势。

5.4 MesaTEE PSI 的性能优势

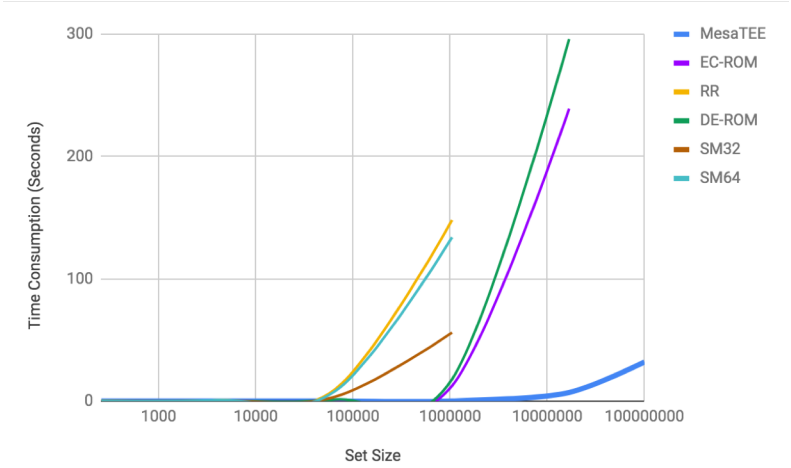

图 4: MesaTEE PSI 的优越性能。相比传统 PSI，MesaTEE PSI 具有 *60* 倍以上的性能优势。

为了全面展示 MesaTEE PSI 的性能优势，我们和一些高性能的传统 PSI 做了性能对比。实验搭建在 LAN 的环境下面。LAN 的传输速度 10Gbps。参与方进行单线程的 PSI 运算。参与方机器配置：Intel(R) Xeon(R) CPU E3-1280 v6 @ 3.90GHz 和 128MB 的 SGX enclave 内存。对于多线程情况，MesaTEE 方案可以很容易开启多个可信节点进行扩展。

对比结果总结在图4里面。图4的横轴是求交集合的大小，纵轴是求交所需时间。从图中我们可以观察到有些传统 PSI 方案只能在很小的 dataset 上面进行 PSI 运算（比如 RR [54]，SM32，SM64 [55] 等）。这样的方案往往只能处理百万以内的数据，超过这个数据量就无法完成整个运算。对于 EC-ROM 和 DE-ROM 方案 [55]，他们客服了其他传统 PSI 的瓶颈，可以处理较大数据集（比如超过千万的数据集）。但是和 MesaTEE PSI 相比较，而然具有较大的性能不足（见图4）。从图中结果我们可以看到，MesaTEE PSI 比目前最快的传统 PSI 方案还要快 *60* 倍以上。而且，MesaTEE PSI 可以做流式求交，并且可以分布式并行处理。这样性能差距还会进一步拉大。MesaTEE PSI 可以自

由的从 2 方求交、3 方求交扩展到任意多方求交，而且代价基本不变。事实上，参与者越多，要分析的数据集越大，与传统方法相比，MesaTEE PSI 会显示出更大的性能优势，还可以解决传统 PSI 无法完成的复杂计算场景问题。

参考文献

[1] Asharov, G., Lindell, Y., Schneider, T., and Zohner, M. More efficient oblivious transfer and extensions for faster secure computation. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (2013), ACM, pp. 535–548.

[2] Beaver, D. Correlated pseudorandomness and the complexity of private computations. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (1996), ACM, pp. 479–488.

[3] Blakley, G. R., et al. Safeguarding cryptographic keys. In *Proceedings of the national computer conference* (1979), vol. 48, pp. 313–317.

[4] Bogdanov, D., Laur, S., and Willemson, J. Sharemind: A framework for fast privacy-preserving computations. In *European Symposium on Research in Computer Security* (2008), Springer, pp. 192–206.

[5] Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT) 6*, 3 (2014), 13.

[6] Brakerski, Z., and Vaikuntanathan, V. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on Computing 43*, 2 (2014), 831–871.

[7] Burkhart, M., Strasser, M., Many, D., and Dimitropoulos, X. Sepia: Privacy-preserving aggregation of multi-domain network events and statistics. *Network 1*, 101101 (2010).

[8] Büscher, N., and Katzenbeisser, S. Faster secure computation through automatic parallelization. In *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015.* (2015), J. Jung and T. Holz, Eds., USENIX Association, pp. 531–546.

[9] Chen, H., Huang, Z., Laine, K., and Rindal, P. Labeled psi from fully homomorphic encryption with malicious security. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (2018), ACM, pp. 1223–1237.

[10] Chen, H., Laine, K., and Rindal, P. Fast private set intersection from homomorphic encryption. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), ACM, pp. 1243–1255.

[11] Chen, H., Laine, K., and Rindal, P. Fast private set intersection from homomorphic encryption. Cryptology ePrint Archive, Report 2017/299, 2017. `https://eprint.iacr.org/2017/299`.

[12] Ciampi, M., and Orlandi, C. Combining private set-intersection with secure two-party computation. *IACR ePrint 105* (2018), 2018.

[13] Cristofaro, E. D., and Tsudik, G. Practical private set intersection protocols with linear complexity. In Sion [59], pp. 143–159.

[14] Cristofaro, E. D., and Tsudik, G. Experimenting with fast private set intersection. In *Trust and Trustworthy Computing - 5th International Conference, TRUST 2012, Vienna, Austria, June 13-15, 2012. Proceedings* (2012), S. Katzenbeisser, E. R. Weippl, L. J. Camp, M. Volkamer, M. K. Reiter, and X. Zhang, Eds., vol. 7344 of *Lecture Notes in Computer Science*, Springer, pp. 55–73.

[15] Damgård, I., Geisler, M., Krøigaard, M., and Nielsen, J. B. Asynchronous multiparty computation: Theory and implementation. In *International Workshop on Public Key Cryptography* (2009), Springer, pp. 160–179.

[16] Debnath, S. K., and Dutta, R. Secure and efficient private set intersection cardinality using bloom filter. In *Information Security - 18th International Conference, ISC 2015, Trondheim, Norway, September 9-11, 2015, Proceedings* (2015), J. López and C. J. Mitchell, Eds., vol. 9290 of *Lecture Notes in Computer Science*, Springer, pp. 209–226.

[17] Demmler, D., Schneider, T., and Zohner, M. ABY - A framework for efficient mixed-protocol secure two-party computation. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015* (2015), The Internet Society.

[18] Demmler, D., Schneider, T., and Zohner, M. Aby-a framework for efficient mixed-protocol secure two-party computation. In *NDSS* (2015).

[19] Dong, C., Chen, L., and Wen, Z. When private set intersection meets big data: an efficient and scalable protocol. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013* (2013), A. Sadeghi, V. D. Gligor, and M. Yung, Eds., ACM, pp. 789–800.

[20] ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory 31*, 4 (1985), 469–472.

[21] Even, S., Goldreich, O., and Lempel, A. A randomized protocol for signing contracts. *Communications of the ACM 28*, 6 (1985), 637–647.

[22] Freedman, M. J., Hazay, C., Nissim, K., and Pinkas, B. Efficient set intersection with simulation-based security. *Journal of Cryptology 29*, 1 (2016), 115–155.

[23] Freedman, M. J., Ishai, Y., Pinkas, B., and Reingold, O. Keyword search and oblivious pseudorandom functions. In *Theory of Cryptography Conference* (2005), Springer, pp. 303–324.

[24] Freedman, M. J., Nissim, K., and Pinkas, B. Efficient private matching and set intersection. In *International conference on the theory and applications of cryptographic techniques* (2004), Springer, pp. 1–19.

[25] Gentry, C., et al. Fully homomorphic encryption using ideal lattices. In *Stoc* (2009), vol. 9, pp. 169–178.

[26] Gentry, C., Sahai, A., and Waters, B. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Annual Cryptology Conference* (2013), Springer, pp. 75–92.

[27] Goldreich, O., Micali, S., and Wigderson, A. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing* (1987), ACM, pp. 218–229.

[28] Goldreich, O., Micali, S., and Wigderson, A. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA* (1987), A. V. Aho, Ed., ACM, pp. 218–229.

[29] Huang, Y., Evans, D., and Katz, J. Private set intersection: Are garbled circuits better than custom protocols? In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012* (2012), The Internet Society.

[30] Huang, Y., Evans, D., Katz, J., and Malka, L. Faster secure two-party computation using garbled circuits. In *20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings* (2011), USENIX Association.

[31] Huberman, B. A., Franklin, M., and Hogg, T. Enhancing privacy and trust in electronic communities. In *Proceedings of the 1st ACM conference on Electronic commerce* (1999), ACM, pp. 78–86.

[32] Impagliazzo, R., and Rudich, S. Limits on the provable consequences of one-way permutations. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing* (1989), ACM, pp. 44–61.

[33] Ishai, Y., Kilian, J., Nissim, K., and Petrank, E. Extending oblivious transfers efficiently. In *Annual International Cryptology Conference* (2003), Springer, pp. 145–161.

[34] Kamara, S., Mohassel, P., Raykova, M., and Sadeghian, S. S. Scaling private set intersection to billion-element sets. In *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers* (2014), N. Christin and R. Safavi-Naini, Eds., vol. 8437 of *Lecture Notes in Computer Science*, Springer, pp. 195–215.

[35] Kamara, S., Mohassel, P., and Riva, B. Salus: a system for server-aided secure function evaluation. In *Proceedings of the 2012 ACM conference on Computer and communications security* (2012), ACM, pp. 797–808.

[36] Kissner, L., and Song, D. X. Privacy-preserving set operations. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings* (2005), V. Shoup, Ed., vol. 3621 of *Lecture Notes in Computer Science*, Springer, pp. 241–257.

[37] Kolesnikov, V., and Kumaresan, R. Improved ot extension for transferring short secrets. In *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, pp. 54–70.

[38] Kolesnikov, V., Kumaresan, R., Rosulek, M., and Trieu, N. Efficient batched oblivious prf with applications to private set intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016), ACM, pp. 818–829.

[39] Kolesnikov, V., Kumaresan, R., Rosulek, M., and Trieu, N. Efficient batched oblivious prf with applications to private set intersection. Cryptology ePrint Archive, Report 2016/799, 2016. `https://eprint.iacr.org/2016/799`.

[40] Lambæk, M. Breaking and fixing private set intersection protocols. *IACR Cryptology ePrint Archive 2016* (2016), 665.

[41] Liu, C., Wang, X. S., Nayak, K., Huang, Y., and Shi, E. Oblivm: A programming framework for secure computation. In *Security and Privacy (SP), 2015 IEEE Symposium on* (2015), IEEE, pp. 359–376.

[42] Malkhi, D., Nisan, N., Pinkas, B., Sella, Y., et al. Fairplay-secure two-party computation system. In *USENIX Security Symposium* (2004), vol. 4, San Diego, CA, USA, p. 9.

[43] Meadows, C. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *A More Efficient Cryptographic Matchmaking Protocol for Use in the Absence of a Continuously Available Third Party* (1986), IEEE, pp. 134–134.

[44] Naor, M., Pinkas, B., and Sumner, R. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM conference on Electronic commerce* (1999), ACM, pp. 129–139.

[45] Orrù, M., Orsini, E., and Scholl, P. Actively secure 1-out-of-n OT extension with application to private set intersection. In *Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings* (2017), H. Handschuh, Ed., vol. 10159 of *Lecture Notes in Computer Science*, Springer, pp. 381–396.

[46] Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques* (1999), Springer, pp. 223–238.

[47] Pinkas, B. Fair secure two-party computation. In *International Conference on the Theory and Applications of Cryptographic Techniques* (2003), Springer, pp. 87–105.

[48] Pinkas, B., Schneider, T., Segev, G., and Zohner, M. Phasing: Private set intersection using permutation-based hashing. In *USENIX Security Symposium* (2015), vol. 15, pp. 515–530.

[49] Pinkas, B., Schneider, T., Smart, N. P., and Williams, S. C. Secure two-party computation is practical. In *International Conference on the Theory and Application of Cryptology and Information Security* (2009), Springer, pp. 250–267.

[50] Pinkas, B., Schneider, T., and Zohner, M. Faster private set intersection based on ot extension. In *USENIX Security Symposium* (2014), vol. 14, pp. 797–812.

[51] Pinkas, B., Schneider, T., and Zohner, M. Scalable private set intersection based on ot extension. Cryptology ePrint Archive, Report 2016/930, 2016. `https://eprint.iacr.org/2016/930`.

[52] Pinkas, B., Schneider, T., and Zohner, M. Scalable private set intersection based on ot extension. *ACM Transactions on Privacy and Security (TOPS) 21*, 2 (2018), 7.

[53] Rindal, P., and Rosulek, M. Faster malicious 2-party secure computation with online/offline dual execution. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.* (2016), T. Holz and S. Savage, Eds., USENIX Association, pp. 297–314.

[54] Rindal, P., and Rosulek, M. Improved private set intersection against malicious adversaries. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I* (2017), J. Coron and J. B. Nielsen, Eds., vol. 10210 of *Lecture Notes in Computer Science*, pp. 235–259.

[55] Rindal, P., and Rosulek, M. Malicious-secure private set intersection via dual execution. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), ACM, pp. 1229–1242.

[56] Schneider, T., and Zohner, M. GMW vs. yao? efficient secure two-party computation with low depth circuits. In *Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers* (2013), A. Sadeghi, Ed., vol. 7859 of *Lecture Notes in Computer Science*, Springer, pp. 275–292.

[57] Shamir, A. How to share a secret. *Communications of the ACM 22*, 11 (1979), 612–613.

[58] Shamir, A. On the power of commutativity in cryptography. In *International Colloquium on Automata, Languages, and Programming* (1980), Springer, pp. 582–595.

[59] Sion, R., Ed. *Financial Cryptography and Data Security, 14th International Conference, FC 2010, Tenerife, Canary Islands, Spain, January 25-28, 2010, Revised Selected Papers* (2010), vol. 6052 of *Lecture Notes in Computer Science*, Springer.

[60] Yao, A. C.-C. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on* (1986), IEEE, pp. 162–167.

[61] Zahur, S., Rosulek, M., and Evans, D. Two halves make a whole. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2015), Springer, pp. 220–250.

[62] 申立艳, 陈小军, 时金桥, and 胡兰兰. 隐私保护集合交集计算技术研究综述. *计算机研究与发展 54*, 10 (2017), 2153–2169.

[63] 蒋瀚, and 徐秋亮. 实用安全多方计算协议关键技术研究进展. *计算机研究与发展 52*, 10 (2015), 2247–2257.