MPC 介绍

MPC 定义，安全模型

张聪
zhangcong@iie.ac.cn

中国科学院信息工程研究所国家重点实验室

2023 年 11 月 2 日

1 MPC 定义

2 安全模型

3 一些细节

4 参考文献

1 MPC 定义

2 安全模型

3 一些细节

4 参考文献

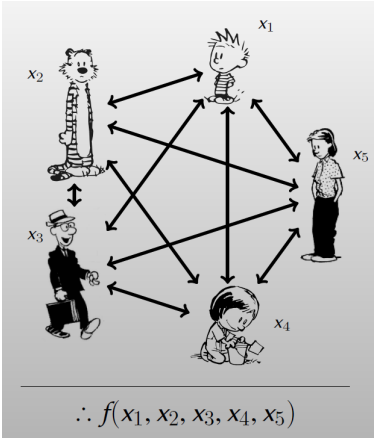
MPC 定义

安全多方计算 (Secure Multi-Party Computation, SMPC/MPC) 问题最早来源于姚期智先生在 1982 年 [Yao82] 里提出的姚氏百万富翁问题，两个百万富翁 Alice 和 Bob 在无任何可信第三方，同时不暴露自己的财产的情况下，希望得出谁更富有的结论。为了解决这一问题，姚期智先生于 1986 年 [Yao86] 提出了基于混淆电路的通用解决方案，进一步验证了多方安全计算的通用可行性，同时也奠定了多方安全计算的理论基础。此后，经 Oded Goldreich、Shafi Goldwasser 等学者进一步的研究和创新，多方安全计算逐渐发展成为现代密码学的一个重要分支。

MPC 定义

一般场景：

- 各方拥有私有输入 x_i.
- 各方得到输出结果 $f(\{x_i\})$.
- 各方只能得到输出结果.



1 MPC 定义

2 安全模型
- 安全性需求
- 真实-理想范式 (Real-Ideal Paradigm)

3 一些细节

4 参考文献

1 MPC 定义

2 安全模型
- 安全性需求
- 真实-理想范式 (Real-Ideal Paradigm)

3 一些细节

4 参考文献

基本目标

我们考虑的情况是一个敌手 $\mathcal{A}$ 控制某些参与方的子集并希望攻击协议执行。在敌手控制下的各方被称为 corrupted party，并遵循敌手的指示。安全协议应能抵御任何敌手的攻击，为了正式声明和证明协议是安全的，需要对多方计算的安全性进行精确定义。一些需要满足的性质如下：

- 隐私性 *(Privacy)*：任何一方都不应该得到超出其规定输出的信息.
- 正确性 *(Correctness)*：保证各方收到的输出是正确的.
- 输入独立性 *(Independence of Inputs)*：腐化方的输入必须独立于诚实方的输入.
- 保证输出呈递性 *(Guaranteed Output Delivery)*：腐化方不能阻止诚实方得到输出.
- 公平性 *(Fairness)*：当且仅当诚实方得到其输出时，腐化方收到其输出. 保证输出呈递性可以推出公平性，反之不然.
- .....

如何定义安全性

能否将满足上述性质的协议定义为安全?

- 可能遗漏一些重要的性质

如何定义安全性

能否将满足上述性质的协议定义为安全?

- 可能遗漏一些重要的性质
- 不清楚什么时候完整

如何定义安全性

能否将满足上述性质的协议定义为安全?

- 可能遗漏一些重要的性质
- 不清楚什么时候完整
- 定义不够简单

1 MPC 定义

2 安全模型
- 安全性需求
- 真实-理想范式 (Real-Ideal Paradigm)

3 一些细节

4 参考文献

真实-理想范式 (Real-Ideal Paradigm)

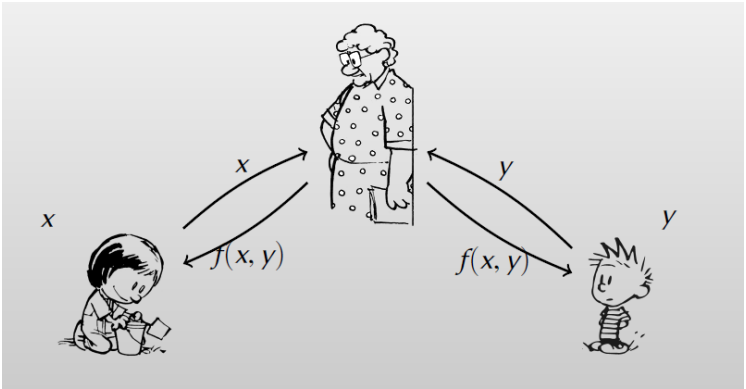
理想世界：一个外部的可信第三方愿意帮助各方进行计算。在这样的世界中，各方可以简单地将其输入发送给可信方，然后可信方计算所需的函数并将其指定的输出返回给各方。

注：安全性需要的性质在理想世界中一定成立。

真实世界：无可信第三方，各方通过执行协议 π 进行交互，最终得到输出。

安全性定义思想：如果对协议的任何攻击在由可信第三方计算函数的理想世界中存在等效攻击，则协议是安全的。

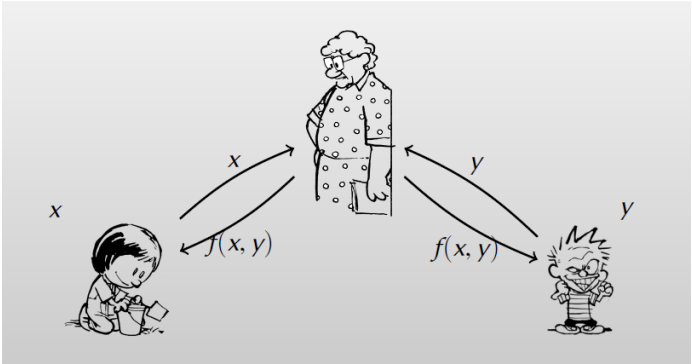
真实-理想范式 (Real-Ideal Paradigm)



真实-理想范式 (Real-Ideal Paradigm)

敌手可以做的：

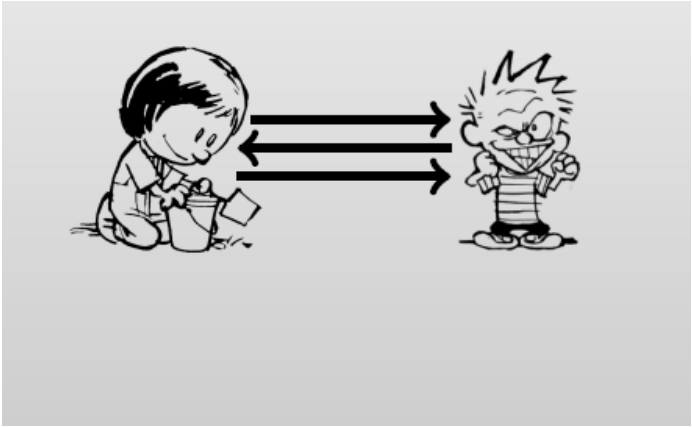
- 选择任意输入 y(独立于 x).
- 只能得到 $f(x, y)$.
- 令诚实方得到 $f(x, y)$.



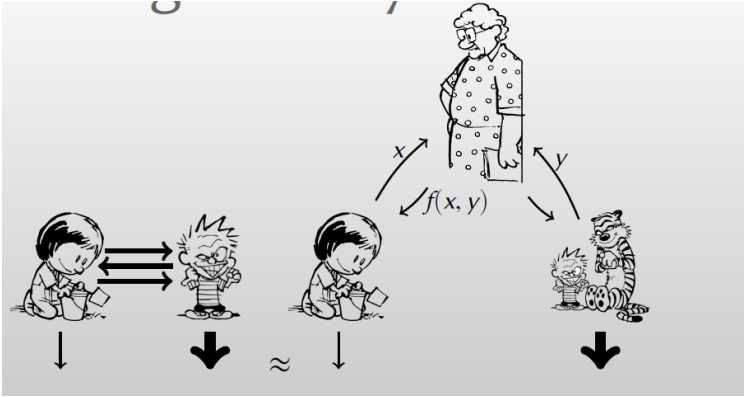
真实-理想范式 (Real-Ideal Paradigm)

敌手可能可以做的:

- 学到诚实方输入的额外信息.
- 干扰诚实方的输出.

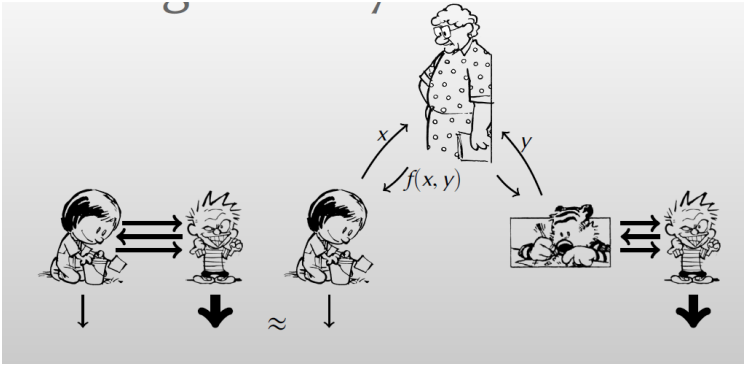


真实-理想范式 (Real-Ideal Paradigm)



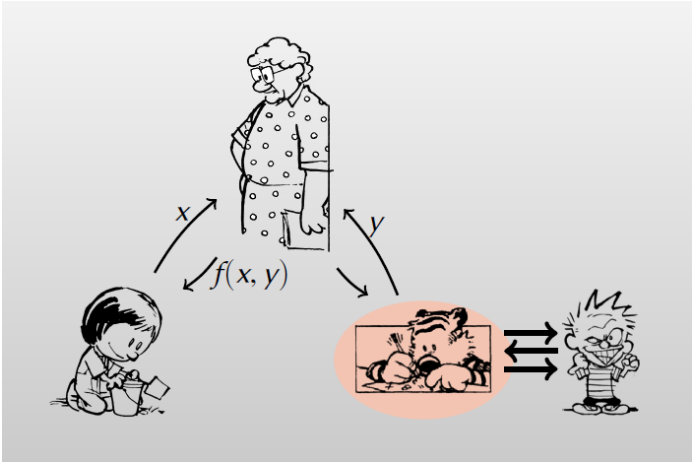
安全性定义：对每个真实世界敌手 $\mathcal{A}$，均存在理想世界的敌手 $\mathcal{S}$，使得两个世界中的联合分布 (诚实者输出，敌手输出) 不可区分。

真实-理想范式 (Real-Ideal Paradigm)



安全性定义：也即，存在理想世界中的模拟器 S，可以模拟真实世界中的交互。

真实-理想范式 (Real-Ideal Paradigm)



目标：构造模拟器 $\mathcal{S}$

半诚实安全

半诚实 (semi-honest/honest-but-curious/passive)：即敌手按照协议规定执行，想从协议的执行副本中获得额外的信息.
一个参与方的 View 由他的输入，随机带，以及在协议中收到的所有消息组成 (注，不包括输出，但输出可以由 view 计算得到)。敌手 $\mathcal{A}$ 的 view 由他控制的所有 corrupted party 的 view 组成。敌手运行协议中学到的任何东西都一定是其 view 的高效可计算函数。也就是说，在不失一般性的情况下，我们只需要考虑一种“攻击”，敌手只需输出其全部 view。
令 π 是计算理想功能 $\mathcal{F}$ 的协议，令 C 表示 corrupted party 集合，定义分布如下：

- $\mathsf{Real}_{\pi,\mathcal{A}}(\kappa, C; x_1, \ldots, x_n)$: 每方 P_i 使用 x_i 作为输入，诚实地执行协议，令 V_i 表示 P_i 的 view，y_i 表示 P_i 的输出.
  输出 $(\{V_i | i \in C\}, (y_1, \ldots, y_n))$
- $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}(\kappa, C; x_1, \ldots, x_n)$: 计算 $(y_1, \ldots, y_n) \leftarrow \mathcal{F}(x_1, \ldots, x_n)$
  输出 $(\mathcal{S}(C, \{(x_i, y_i) | i \in C\}), (y_1, \ldots, y_n))$

半诚实安全

定义 (半诚实安全)

我们称协议 π 半诚实安全地实现功能 $\mathcal{F}$，如果存在一个模拟器 $\mathcal{S}$，使得对所有 corrupted party 集合 $C \subset [n]$，所有的输入 $x_1, \ldots, x_n$，有
$\mathsf{Real}_{\pi,\mathcal{A}}(\kappa, C; x_1, \ldots, x_n) \approx \mathsf{Ideal}_{\mathcal{F},\mathcal{S}}(\kappa, C; x_1, \ldots, x_n)$

恶意安全

恶意 (malicious/active)：敌手可以任意偏离协议，此时安全的定义思想依然是理想和真实不可区分，但是需要额外考虑两点：

1. 对诚实方输出的影响。由于敌手可以任意偏离协议，因此有可能通过发送不同的消息来影响诚实方的输出。但是在半诚实的定义中没有这个问题，因为半诚实情况下，诚实方的输出不依赖于敌手。除此之外，敌手的输出没有任何保证，因为一个恶意的敌手可以输出任意它想输出的。
2. 提取输入。敌手在真实世界中的输入并没有定义，因为敌手可以任意偏离协议，也就意味着敌手可以将任意值作为输入。既然敌手在真实世界的输入无法保证，如何模拟敌手在理想世界中给理想功能的输入就成了一个问题。定义安全的思想是，任何敌手在真实协议中可以做的攻击，在理想世界中也可以实现。如果真实世界中敌手通过选择 corrupt 方的输入来进行攻击，则模拟器 (即理想敌手) 也需要通过自己选择 corrupt 方的输入在理想世界中进行攻击，但是如何在理想世界中选择对应真实世界敌手的输入，这就需要模拟器对真实世界中敌手的输入进行提取。提取过程一般是通过对真实世界敌手进行黑盒访问来进行。

恶意安全

接下来同样考虑两个随机变量：

- $\mathsf{Real}_{\pi,\mathcal{A}}(\kappa, C; \{x_i | i \notin C\})$：每个诚实方 P_i 使用 x_i 作为输入，诚实地执行协议，corrupted parties 的输入由敌手 $\mathcal{A}$ 选取，令 y_i 表示诚实方 P_i 的输出，V_i 表示 P_i 的 view.
  输出 $(\{V_i | i \in C\}, \{y_i | i \notin C\})$
- $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}(\kappa, C; \{x_i | i \notin C\})$：运行模拟器 $\mathcal{S}$ 直到模拟器输出 $\{x_i | i \in C\}$，计算 $(y_1, \ldots, y_n) \leftarrow f(x_1, \ldots, x_n)$，然后将 $\{y_i | i \in C\}$ 给模拟器 $\mathcal{S}$，用 V^* 表示模拟器的最终输出 (即模拟的敌手 view)，
  输出 $(V^*, \{y_i | i \notin C\})$

注：这里可以把模拟器看成两个算法 $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$，其中 $\mathcal{S}_1$ 输入安全参数 κ，输出 $\{x_i | i \in C\}$ 和任意的中间状态 Σ，$\mathcal{S}_2$ 输入 $(\Sigma, \{y_i | i \in C\})$，输出 V^*。

恶意安全

则可以定义恶意安全如下：

定义 (恶意安全)

我们称协议 π 恶意安全地实现功能 $\mathcal{F}$，如果对任意敌手 $\mathcal{A}$，存在一个模拟器 $\mathcal{S}$，使得对所有 corrupted party 集合 $C \subset [n]$，所有诚实方的输入 $\{x_i | i \notin C\}$，有 $\mathsf{Real}_{\pi,\mathcal{A}}(\kappa, C; \{x_i | i \notin C\}) \approx \mathsf{Ideal}_{\mathcal{F},\mathcal{S}}(\kappa, C; \{x_i | i \notin C\})$

注：敌手在恶意模型下不需要考虑输入，因为敌手可以任意选择输入，即使给了输入 x_i，敌手也可能不用这个输入。

可终止安全性 (security with abort)

在某些情况下，定义会放松到不满足保证输出呈递性以及公平性，当这些被排除在外时所达到的安全级别被称为“可终止安全性”(security with abort)，其结果是敌手可能能够获得输出，而诚实方无法获得输出。使用这种放松的主要原因有两个。首先，在某些情况下，不可能实现公平（例如，不可能为两方实现公平的投币协议 [Cle86]）第二，在某些情况下，当不保证公平性时，可以得到更高效的协议。因此，如果应用程序不需要公平性（尤其是在只有一方接收输出的情况下），这种放松是有帮助的。当考虑可终止安全性时，一般需要对理想功能做一些改动：首先，允许理想功能了解腐化方的身份。在所有各方提供输入后，该功能计算输出并仅将输出传递给腐化方。然后，该功能等待敌手发出“交付 (deliver)”或“中止 (abort)”命令。收到“交付”后，该功能将输出发给所有诚实方。收到“中止”后，该功能将提供中止输出 $\perp$ 给所有诚实方。

顺序模块组合 (sequential modular composition)

独立场景 (stand-alone setting):

- MPC 协议可以作为另一协议的子协议运行.
- 可以在 MPC 协议之前和之后发送任意其他消息.
- MPC 协议本身必须在没有并行发送任何其他消息的情况下运行.

[Can00] 的顺序模块组合定理 (sequential modular composition theorem) 指出，在这种情形下，MPC 协议的行为确实类似于可信第三方执行的计算。

顺序模块组合 (sequential modular composition)

设 π_f 是一个安全计算 f 的协议，它使用子协议 π_g 来计算 g。考虑混合模型中 f 的执行，其中使用可信第三方来理想地计算 g（而不是让各方运行真实子协议 π_g）。顺序模块组合定理通过以下方法进行安全性的模块化分析：首先证明 π_g 的安全性，然后在有可信第三方计算 g 的模型 (也称为 g-hybrid model) 中证明 π_f 的安全性.

通用组合 (universal composability,UC) 安全性

在某些情况下，MPC 协议与自身的其他实例、其他 MPC 协议和其他不安全协议同时运行。在这些情况下，在上述 stand-alone 情形下证明安全的协议实际上可能不会保持安全。此时需要用到通用组合 (universal composability,UC) 安全性 [Can01]。
UC 框架通过一个称为环境 (environment) 的附加实体 $\mathcal{Z}$ 来增强安全模型，且它同时包含在理想世界和真实世界中。环境的目的是探测协议执行的“上下文”(context)（例如，所考虑的协议在一些更大的调用协议中作为一个小步骤被调用）。环境为诚实方选择输入，并得到他们的输出。它也可能与敌手进行任意交互。
环境 $\mathcal{Z}$ 的“目标”是确定它是在现实世界还是理想世界中。以前，我们通过要求真实 View 和理想 View 不可区分来定义安全性。在 UC 场景中，我们一般考虑将这个 View 的区分器和环境合并。因此，环境的最终输出可以只是一个比特，它可以被解释为环境对它是在真实世界还是理想世界中实例化的“猜测”。

通用组合 (universal composability,UC) 安全性

考虑两个随机变量：

- $\mathsf{Real}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa)$：与敌手 $\mathcal{A}$ 和环境 $\mathcal{Z}$ 的交互，当环境 $\mathcal{Z}$ 为诚实方生成输入时，诚实方运行协议 π，并将输出给 $\mathcal{Z}$。最后 $\mathcal{Z}$ 输出一个比特，作为 $\mathsf{Real}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa)$ 的输出。
- $\mathsf{Ideal}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\kappa)$：与敌手 $\mathcal{S}$ 和环境 $\mathcal{Z}$ 的交互，当环境 $\mathcal{Z}$ 为诚实方生成输入时，输入直接传递给理想功能 $\mathcal{F}$，理想功能计算输出后直接将诚实方的输出发给 $\mathcal{Z}$。最后 $\mathcal{Z}$ 输出一个比特，作为 $\mathsf{Ideal}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\kappa)$ 的输出。

定义 (UC 安全)

我们称协议 π UC 安全地实现功能 $\mathcal{F}$，如果对任意敌手 $\mathcal{A}$，存在一个模拟器 $\mathcal{S}$，对所有环境 $\mathcal{Z}$，有

$\mathsf{Real}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa) \approx \mathsf{Ideal}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(\kappa)$

通用组合 (universal composability,UC) 安全性

定义 (Start synchronization，[KLR10])

We say that a protocol ρ has start synchronization if it begins with the following steps for all parties:

1. Send "begin ρ"to all parties.
2. Wait until "begin ρ"messages are received from all parties. Then, continue the execution of ρ.

定理 ([KLR10],Theorem 1.5)

Every protocol that is secure in the stand-alone model and has ***start synchronization*** *and a* ***straight-line black-box simulator*** *is secure under concurrent general composition.*

其他安全模型

按照敌手行为划分：

- 半诚实/被动 (semi-honest/passive)，即敌手按照协议规定执行，想从协议的执行副本中获得额外的信息
- 恶意/主动 (malicious/active)，即敌手可以任意偏离协议规范执行。
- 隐蔽 (covert)：由 [AL07] 最早提出，即敌手表现为恶意时有一定的概率 ϵ 被诚实方发现，这里 ϵ 被称为威慑因子 (deterrence factor)。当威慑因子为 overwhelming 时，covert 和 malicious 就是等价的。随后 [AO12] 在此基础上提出了更高的安全性，称为公开可验证的隐蔽安全性 (publicly verifiable covert, PVC)，表示敌手一旦被抓到作弊，诚实方可以生成一个证明 π，可以证明敌手作弊，且可以由任何人公开验证。目前 PVC 的后续工作比较多，有 [KM15; Hon+19; DOS20; Fau+21; SSS21] 前两篇是提升效率，后几篇是通用编译器。

其他安全模型

按敌手计算能力划分：

- 信息论/完美/无条件安全 (information theory/perfect/unconditionally security)，即模拟器的输出和真实的 view 同分布。
- 统计安全 (statistical security)，即模拟器的输出和真实的 view 统计不可区分。
- 计算安全 (computational security)，即模拟器的输出和真实的 view 计算不可区分。

其他安全模型

按敌手数量划分：

- 不诚实大多数（dishonest majority）：即敌手最多可以有 $n-1$ 个。用 t 表示敌手数量，则 $t < n$。目前基于 GC，GMW，BMR 框架的协议均满足这一点。两方协议一定是 dishonest majority。
- 诚实大多数（honest majority），即敌手的数量不超过参与方总数的一半，$t < \frac{n}{2}$。主要在 BGW 框架下。有时为了达到更高的安全性或者效率会有 $t < \frac{n}{3}, t < (\frac{1}{2} - \epsilon)n$ 等情况。

其他安全模型

按敌手对输出的控制程度划分：

- 保证输出呈递性 (guaranteed output delivery,GOD)，即敌手无法干扰诚实方得到正确的输出。这个性质是最强的，但并不总是能够保证，只有 honest majority 可以保证 GOD [GSZ20]。
- 公平性 (fairness)，即敌手要么和诚实方同时得到输出，要么同时终止。GOD 可以推出 fairness，但反之不成立。同样地，fairness 也并不总是能够保证，例如永远无法实现公平的投币协议 [Cle86]。
- 可终止安全性 (security with abort)，即敌手可以先得到输出，再由敌手决定是否让诚实方得到输出。这也是目前实际 MPC 构造中最常见的类别。可终止安全性又可以分为下面几类：
  - Security with identifiable abort，即诚实方可以知道敌手身份。
  - Security with unanimous abort，即诚实方可以检测到敌手但不知身份，敌手可以选择让所有诚实方同时终止或输出。
  - Security with selective (non-unanimous) abort，即诚实方要么检测到作弊行为而终止要么输出，敌手可以选择让哪些诚实方输出，哪些诚实方终止。

其他安全模型

按敌手确定时间划分：

- 静态 (static)：即敌手在协议开始前就确定 corrupt 哪些 party。目前绝大部分实用的 MPC 协议都是考虑静态敌手。
- 动态 (adaptive)：即敌手可以在协议开始后再确定 corrupt 哪些 party。这里 corrupt party 一旦被 corrupt 后就一直是 corrupt party。根据敌手是否能读取 corrupt 方过去的 view，又可以分为两类：
  - No erasures model，即敌手在 corrupt 之后可以得到 corrupt party 的整个 view，包括他的输入，随机带，收到的消息。[Can+96; Can00; Can+02]
  - Erasures model，即允许 corrupt party 在被敌手 corrupt 时擦除一些数据。[BH92; Lin09]
- 移动 (proactive/mobile)：即敌手可以只 corrupt 某个 party 一段时间，也就是说，honest party 可能变成 corrupt party，而 corrupt party 也可能变成 honest party。[OY91; CH94]

1 MPC 定义

2 安全模型

3 一些细节

4 参考文献

允许任何输入

尽管真实理想范式提供了一个简单的抽象，但有一个微妙的点有时会被误解。MPC 协议的行为类似于理想的执行；因此，所获得的安全性类似于理想执行的安全性。然而，在理想的执行中，敌手可以输入他们希望的任何值，事实上没有通用的方法来防止这种情况。因此，如果两个人想知道谁的工资更高（除了这一点信息，没有透露任何信息），那么没有什么能阻止他们中的一个人输入最大可能的值作为他们的工资（然后诚实地遵守 MPC 协议本身），结果是他的收入更高。因此，如果应用程序的安全性取决于当事人使用正确的输入，那么必须使用机制来强制执行。例如，可以要求签名输入，并将签名作为 MPC 计算的一部分进行验证。根据具体的协议，这可能会增加显著的成本。

MPC 只保护过程，不保护输出

另一个经常被误解的微妙之处是 MPC 确保了过程的安全，这意味着计算本身不会泄漏任何信息。然而，这并不意味着正在计算的函数的输出不会显示敏感信息。举个极端的例子，假设两个人计算他们的平均工资。的确，除了平均值之外，什么都不会“泄漏”，但考虑到一个人自己的工资和两个工资的平均值，他们可以得出另一个人的确切工资。因此，仅仅使用 MPC 并不意味着所有隐私问题都得到解决。相反，MPC 确保了计算过程的安全，由于隐私问题，哪些功能应该计算，哪些功能不应该计算，这一问题仍需解决。在某些情况下，如门限密码，这个问题不是问题（因为密码函数的输出不会泄露密钥，假设它是安全的）。然而，在其他情况下，可能不太清楚。

半诚实和恶意的关系

直觉上如果一个协议是恶意安全的，那么一定是半诚实安全的，但实际上并不是这样。这是因为恶意敌手有随意选择输入的能力，而半诚实敌手没有。Hazay 和 Lindell [HL10a] 给了两个例子。

例 1：布尔 AND，即 $f(x,y)=x\wedge y$，P_1 输入是一个比特 x，P_2 输入是一个比特 y，只有 P_2 拿到输出 $x\wedge y$。协议如下：

1. P_1 将 x 发给 P_2.
2. P_2 输出 $x\wedge y$.

证明：我们分别考虑 corrupted P_1 的情况和 corrupted P_2 的情况。

对 corrupted P_1，则模拟器 $\mathcal{S}$ 从 $\mathcal{A}$ 接收其在协议中发送给 P_2 的比特。这个比特完全确定了 P_1 的输入，因此 $\mathcal{S}$ 仅将其发送给理想功能，从而完成模拟。

对 corrupted P_2，$\mathcal{S}$ 将输入 1 发送给理想功能，并收到输出比特 b。由于 $b=x\wedge 1=x$，b 是理想模型中诚实 P_1 的输入。因此，模拟器 $\mathcal{S}$ 仅将比特 $x=b$ 交给 $\mathcal{A}$，作为 $\mathcal{A}$ 在实际执行中从诚实 P_1 接收的值。很明显，这里的模拟是完美的.

半诚实和恶意的关系

但上述协议并不是半诚实的，分析如下：
考虑 corrupted P_2 的情况，模拟器 $\mathcal{S}$ 收到 y 和 $x \wedge y$，目标是生成 P_2 的 View。注意到，P_2 的 View 包含 P_1 在协议中发给 P_2 的值 x。如果 $y = 0$ 并且 x 是随机的，那么 $\mathcal{S}$ 无法以大于 $1/2$ 的概率猜测 x 的值。因此，在敌手半诚实的情况下，协议是不安全的。

半诚实和恶意的关系

例 2：PSU，即 $f(X, Y) = X \cup Y$，P_1 输入是一个集合 X，P_2 输入是一个集合 Y，只有 P_2 拿到输出 $X \cup Y$。协议如下：

1. P_1 将 X 发给 P_2.
2. P_2 输出 $X \cup Y$.

同样的，恶意的 P_2 可以用随机选择的集合 Y' 作为输入，此时随机集合 $Y' \cap X = \varnothing$ 的概率是 overwhelming 的，因此 P_2 可以得到 P_1 的全部输入。

半诚实和恶意的关系

定义 (增强半诚实模型 (augmented semi-honest model), [Gol04])

设 π 是一个两方协议。增强的半诚实行为是一种（可行的）策略，它满足以下条件：

- 进入执行（这是唯一不同于标准半诚实行为的定义）：根据其初始输入，用 u 表示，一方可以在执行 π 的任何步骤之前中止。否则，它可以用任何输入 $u' \in \{0,1\}^{|u|}$ 执行协议。从这里开始，u' 固定下来不再改变。
- 正确的随机带的选择：参与方在指定长度的所有字符串中选择 π 中要使用的随机带。也就是说，随机磁带的选择与 π 所指定的完全相同。
- 正确的消息传输或中止：在 π 的每个步骤中，根据执行到目前为止的 View，一方可以中止或按照 π 的规定发送消息。我们强调，消息是根据输入 u'、所选随机带和到目前为止接收到的所有消息按 π 的规定计算的。
- 输出：在交互结束时，参与方根据其协议中的 View 计算输出。我们强调，该 View 由初始输入 u、选定的随机带和到目前为止收到的所有消息组成。

半诚实和恶意的关系

定理 ([HL10a])

设 π 是一个在恶意敌手模型下安全计算函数 f 的协议。则 π 在增强的半诚实敌手模型下安全地计算 f。

确定性函数和概率函数的安全性区别

回忆之前半诚实协议的安全性定义1，在一些文章中，我们其实会发现用的是下面更简单的定义 (两方情况):

定义 (半诚实安全)

设 $f = (f_1, f_2)$ 是确定性函数。我们说在半诚实模型下，协议 π 安全地计算 $f(x, y)$，如果存在 PPT 的 S_1, S_2 使得

$$\{S_1(x, f_1(x, y))\}_{x,y\in\{0,1\}^*} \equiv^c \{View_1^\pi(x, y)\}_{x,y\in\{0,1\}^*}$$
$$\{S_2(y, f_2(x, y))\}_{x,y\in\{0,1\}^*} \equiv^c \{View_2^\pi(x, y)\}_{x,y\in\{0,1\}^*}$$

注意，本定义中，对需要安全计算的函数有一个限制，即要求 f 是确定性函数。如果 f 是一个概率函数，即 f 的输出服从某个分布，这样的话上面的定义就不够用了。考虑下面的例子。

确定性函数和概率函数的安全性区别

- 例 1(来自 [Lin16]):
  考虑计算的函数为 $f(1^\kappa, 1^\kappa) \to (b, \perp)$。即两方都没有输入，$P_1$ 输出一个随机比特 b，P_2 没有输出。我们对安全性的要求是，P_2 不能得到除了自己的输入输出之外的其他信息，特别地，P_2 不能得到 b 的信息。考虑下面的协议:
  1. P_1 随机选择一个比特 $b \leftarrow \{0,1\}$。
  2. P_1 把 b 发给 P_2。
  3. P_1 输出 b，P_2 不输出。

显然上述协议是不安全的，因为 P_2 直接从 P_1 那里得到了 b 的全部信息。但是如果按照之前的定义，我们可以构造模拟器如下,

确定性函数和概率函数的安全性区别

$\mathcal{S}_1(1^\kappa, b)$：随机选择一个随机带 r'，输出 $(1^\kappa, r')$。
真实的 $View_1^\pi(1^\kappa, 1^\kappa) = (1^\kappa, r)$，同分布。
$\mathcal{S}_2(1^\kappa, \perp)$：随机选择一个随机带 r'，随机选择一个比特 $b' \leftarrow \{0,1\}$，输出 $(1^\kappa, r', b')$。
真实的 $View_2^\pi(1^\kappa, 1^\kappa) = (1^\kappa, r, b)$，这里 r 和 b 是互相独立的，因此同分布。
因此可以看到，虽然上述协议在常识上是完全不安全的，但是却可以通过定义证明它是半诚实安全的。问题出在哪里呢？问题在于，P_2 在 view 中看到的消息包含了 P_1 的随机输出，这说明 P_2 的 view 的分布与 P_1 的输出的分布是相关的。但是按照原来的定义，模拟器不需要管 P_1 的输出是什么，只需要模拟 P_2 的 view 即可，也即，模拟器模拟的分布与 P_1 的输出分布是独立的，这显然是有问题的。因此考虑修改定义，使得模拟器需要模拟的除了 view 之外，还需要加上一个限制，即模拟的 view 与输出的联合分布与真实的情况相同。

确定性函数和概率函数的安全性区别

用 $output_i^\pi(x,y)$ 表示第 i 方在执行协议 π 时的输出，$i \in \{1,2\}$。令 $output^\pi(x,y) = (output_1^\pi(x,y), output_2^\pi(x,y))$。考虑修改定义如下：

定义 (半诚实安全)

设 $f = (f_1, f_2)$ 是一个函数（确定/随机）。我们说在半诚实模型下，协议 π 安全地计算 $f(x,y)$，如果存在 PPT 的 S_1, S_2 使得

$$\{(S_1(x, f_1(x,y)), f(x,y))\}_{x,y\in\{0,1\}^*} \equiv^c \{(View_1^\pi(x,y), output^\pi(x,y))\}_{x,y\in\{0,1\}^*}$$
$$\{(S_2(y, f_2(x,y)), f(x,y))\}_{x,y\in\{0,1\}^*} \equiv^c \{(View_2^\pi(x,y), output^\pi(x,y))\}_{x,y\in\{0,1\}^*}$$

注：这里的定义和真实-理想范式是相同的。

确定性函数和概率函数的安全性区别

经过上述修改，要求模拟器输出与函数输出的联合分布，和真实协议执行中的 view 与协议输出的联合分布不可区分。这样就可以避免上述情况。因为经过这样修改之后，我们可以看到之前构造的 $\mathcal{S}_2$ 就不能满足新定义了：

真实的协议中，$(view_2^{\pi}(1^{\kappa}, 1^{\kappa}), output^{\pi}(1^{\kappa}, 1^{\kappa})) = (1^{\kappa}, r, b, (b, \perp))$，这里最后的两个 b 是同一个随机变量，取值相同。

而模拟器的输出是 $(\mathcal{S}_2(1^{\kappa}, \perp), f(1^{\kappa}, 1^{\kappa})) = (1^{\kappa}, r', b', (b, \perp))$。这里最后的 b' 和 b 是互相独立的随机变量。因此无法满足安全性定义。且由于模拟器 $\mathcal{S}_2$ 不知道诚实方的输入输出，即 $\mathcal{S}_2$ 不知道 b，因此无法完成模拟。

确定性函数和概率函数的安全性区别

那么有了新定义之后，是否包含原定义呢，也即，如果 f 是一个确定性函数，这样的定义是否和原来的定义等价？答案是肯定的。

首先给出协议正确性的定义，如果存在可忽略函数 μ，使得对所有 $x, y \in \{0,1\}^*$，所有 κ，有 $Pr[output^{\pi}(x, y, 1^{\kappa}) \neq f(x, y)] \leq \mu(\kappa)$。

当一个确定性函数的协议既满足正确性又满足定义 1 时，则该协议满足定义 2。这是因为区分器可以将 x, y 作为辅助输入，当有了 x, y 之后，区分器可自己计算出 $f(x, y)$，而 $f(x, y)$ 是一个定值，因此

如果 $\{S_1(x, f_1(x, y))\}_{x,y\in\{0,1\}^*} \equiv^c \{View_1^{\pi}(x, y)\}_{x,y\in\{0,1\}^*}$，可以推出 $\{(S_1(x, f_1(x, y)), f(x, y))\}_{x,y\in\{0,1\}^*} \equiv^c \{(View_1^{\pi}(x, y), f(x, y))\}_{x,y\in\{0,1\}^*}$

注：上述关系当 f 是一个概率函数时不成立，因为此时区分器计算 f 用到的随机数和给模拟器的 f_1 不是用同一个随机数计算的。

再由正确性，

$\{(S_1(x, f_1(x, y)), f(x, y))\}_{x,y\in\{0,1\}^*} \equiv^c \{(View_1^{\pi}(x, y), f(x, y))\}_{x,y\in\{0,1\}^*}$ 可以推出 $\{(S_1(x, f_1(x, y)), f(x, y))\}_{x,y\in\{0,1\}^*} \equiv^c \{(View_1^{\pi}(x, y), output^{\pi}(x, y))\}_{x,y\in\{0,1\}^*}$

确定性函数和概率函数协议设计的等价性

虽然确定性函数和概率函数安全性定义上有区别，但实际上如果我们能够针对任意确定性函数设计安全的协议，那么我们就可以得到任意概率函数的安全协议。

令 $f(x, y; r)$ 是概率函数，r 表示 f 用到的随机带，则定义确定性函数 $f'((x, r_1), (y, r_2)) := f(x, y; r_1 \oplus r_2)$。

设 π' 是计算 f' 的安全协议，则构造计算 $f(x, y)$ 的协议 π 如下：

1. P_1 和 P_2 各随机选 r_1 和 r_2。
2. P_1 和 P_2 分别用输入 (x, r_1) 和 (y, r_2) 调用计算 f' 的协议 π'。

则上述协议 π 安全地计算了概率函数 f。证明参考 [Gol04] Section 7.3.

相同输出函数和不同输出函数协议设计的等价性

类似地，相同输出函数和不同输出函数的协议设计也具有等价性。
设 $f(x,y)=(f_1(x,y),f_2(x,y))$ 是任意函数。则定义相同输出函数 $f'((x,r_1),(y,r_2)):=f_1(x,y)\oplus r_1||f_2(x,y)\oplus r_2$。
设 π' 是计算 f' 的安全协议，则构造计算 $f(x,y)$ 的协议 π 如下：

1. P_1 和 P_2 各随机选 r_1 和 r_2。
2. P_1 和 P_2 分别用输入 (x,r_1) 和 (y,r_2) 调用计算 f' 的协议 π'。
3. 令 $w||v$ 表示 π' 的输出，则 P_1 输出 $w\oplus r_1$，P_2 输出 $v\oplus r_2$。

则上述协议 π 安全地计算了函数 f。因为 r_1/r_2 对 P_2/P_1 来说是随机的，模拟器可以直接用随机数模拟。

统计安全参数和计算安全参数

我们分别用 κ 和 λ 表示计算和统计安全参数。计算安全参数决定了对手离线计算所能解决的问题的难度，例如，破坏加密方案。在实践中，通常设置为 128 或 256 这样的值。即使当我们考虑针对计算受限敌手的安全性时，也可能会有一些针对交互式协议的攻击，而离线计算不会使这些攻击变得更容易。例如，协议的交互性质可能只给对手一次违反安全的机会（例如，通过发送具有特殊属性的消息，如预测诚实一方将在下一轮中选择的随机值）。统计安全参数决定了这些攻击的强度。在实践中，通常设置为一个较小的值，如 40 或 80。解释两个安全参数存在的正确方法是，仅在概率为 $2^{-\lambda} + \mathsf{negl}(\kappa)$ 的情况下违反安全性，其中 negl 是可忽略函数，取决于敌手的计算资源。当我们考虑计算上无界的敌手时，要求 $\mathsf{negl}(\kappa) = 0$。

统计安全参数和计算安全参数

定义 ((κ,λ)-不可区分，[LP11])

令 $X=\{X(a,\kappa,\lambda)\}_{\kappa,\lambda\in\mathbb{N};a\in\{0,1\}^*}$ 和 $Y=\{Y(a,\kappa,\lambda)\}_{\kappa,\lambda\in\mathbb{N};a\in\{0,1\}^*}$ 是概率整体，使得对任意 $\kappa,\lambda\in\mathbb{N}$，分布 $X(a,\kappa,\lambda)/Y(a,\kappa,\lambda)$ 在长度是 $\kappa+\lambda$ 的多项式的字符串上取值。我们称整体是 (κ,λ)-不可区分的，记作 $X\overset{\kappa,\lambda}{\equiv}Y$，如果存在一个常数 $0<c\leq 1$ 使得对每个非一致 PPT 区分器 D，每个 $\lambda\in\mathbb{N}$，每个多项式 $p(\cdot)$ 和所有足够大的 $\kappa\in\mathbb{N}$，对每个 $a\in\{0,1\}^*$ 有：

$$|Pr[D(X(a,\kappa,\lambda),a,\kappa,\lambda)=1]-Pr[D(Y(a,\kappa,\lambda),a,\kappa,\lambda)=1]|<\frac{1}{p(\kappa)}+\frac{1}{2^{c\cdot\lambda}}$$

1 MPC 定义

2 安全模型

3 一些细节

4 参考文献

主要参考文献

1. [Gol04] Oded Goldreich. The Foundations of Cryptography - Volume 2: Basic Applications. Cambridge University Press, 2004.
2. [HL10b]Carmit Hazay and Yehuda Lindell. Efficient Secure Two-Party Protocols - Techniques and Constructions. Information Security and Cryptography. Springer, 2010.
3. [Lin16]Yehuda Lindell. “How To Simulate It - A Tutorial on the Simulation Proof Technique”. In: IACR Cryptol. ePrint Arch. (2016), http://eprint.iacr.org/2016/046.
4. [EKR18]David Evans, Vladimir Kolesnikov, and Mike Rosulek. “A Pragmatic Introduction to Secure Multi-Party Computation”. In: Foundations and Trends in Privacy and Security 2.2-3 (2018), pp. 70–246.
5. [Lin20]Yehuda Lindell. Secure Multiparty Computation (MPC). Cryptology ePrint Archive, Paper 2020/300. https://eprint.iacr.org/2020/300. 2020.

参考文献 I

[AO12] Gilad Asharov and Claudio Orlandi. "Calling Out Cheaters: Covert Security with Public Verifiability". In: *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*. 2012, pp. 681–698. DOI: 10.1007/978-3-642-34961-4_41. URL: https://doi.org/10.1007/978-3-642-34961-4_41.

[AL07] Yonatan Aumann and Yehuda Lindell. "Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries". In: *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*. 2007, pp. 137–156. DOI: 10.1007/978-3-540-70936-7_8. URL: https://doi.org/10.1007/978-3-540-70936-7_8.

参考文献 II

[BH92] Donald Beaver and Stuart Haber. "Cryptographic Protocols Provably Secure Against Dynamic Adversaries". In: *Advances in Cryptology - EUROCRYPT '92, Workshop on the Theory and Application of of Cryptographic Techniques, Balatonfüred, Hungary, May 24-28, 1992, Proceedings*. 1992, pp. 307–323. DOI: 10.1007/3-540-47555-9_26. URL: https://doi.org/10.1007/3-540-47555-9_26.

[Can00] Ran Canetti. "Security and Composition of Multiparty Cryptographic Protocols". In: *J. Cryptol.* 13.1 (2000), pp. 143–202. DOI: 10.1007/s001459910006. URL: https://doi.org/10.1007/s001459910006.

参考文献 III

[Can01] Ran Canetti. “Universally Composable Security: A New Paradigm for Cryptographic Protocols”. In: *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*. IEEE Computer Society, 2001, pp. 136–145. DOI: 10.1109/SFCS.2001.959888. URL: https://doi.org/10.1109/SFCS.2001.959888.

[CH94] Ran Canetti and Amir Herzberg. “Maintaining Security in the Presence of Transient Faults”. In: *Advances in Cryptology - CRYPTO ’94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*. 1994, pp. 425–438. DOI: 10.1007/3-540-48658-5_38. URL: https://doi.org/10.1007/3-540-48658-5_38.

参考文献 IV

[Can+96] Ran Canetti et al. “Adaptively Secure Multi-Party Computation”. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*. 1996, pp. 639–648. DOI: 10.1145/237814.238015. URL: https://doi.org/10.1145/237814.238015.

[Can+02] Ran Canetti et al. “Universally composable two-party and multi-party secure computation”. In: *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*. 2002, pp. 494–503. DOI: 10.1145/509907.509980. URL: https://doi.org/10.1145/509907.509980.

参考文献 V

[Cle86] Richard Cleve. “Limits on the Security of Coin Flips when Half the Processors Are Faulty (Extended Abstract)”. In: *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*. 1986, pp. 364–369. DOI: 10.1145/12130.12168. URL: https://doi.org/10.1145/12130.12168.

[DOS20] Ivan Damgård, Claudio Orlandi, and Mark Simkin. “Black-Box Transformations from Passive to Covert Security with Public Verifiability”. In: *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*. 2020, pp. 647–676. DOI: 10.1007/978-3-030-56880-1_23. URL: https://doi.org/10.1007/978-3-030-56880-1_23.

参考文献 VI

[EKR18] David Evans, Vladimir Kolesnikov, and Mike Rosulek. "A Pragmatic Introduction to Secure Multi-Party Computation". In: *Foundations and Trends in Privacy and Security* 2.2-3 (2018), pp. 70–246. DOI: 10.1561/3300000019. URL: https://doi.org/10.1561/3300000019.

[Fau+21] Sebastian Faust et al. "Generic Compiler for Publicly Verifiable Covert Multi-Party Computation". In: *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II*. 2021, pp. 782–811. DOI: 10.1007/978-3-030-77886-6_27. URL: https://doi.org/10.1007/978-3-030-77886-6_27.

参考文献 VII

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2: Basic Applications*. Cambridge University Press, 2004. ISBN: 0-521-83084-2. DOI: 10.1017/CBO9780511721656. URL: http://www.wisdom.weizmann.ac.il/\%7Eoded/foc-vol2.html.

[GSZ20] Vipul Goyal, Yifan Song, and Chenzhi Zhu. "Guaranteed Output Delivery Comes Free in Honest Majority MPC". In: *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*. 2020, pp. 618–646. DOI: 10.1007/978-3-030-56880-1_22. URL: https://doi.org/10.1007/978-3-030-56880-1_22.

参考文献 VIII

[HL10a] Carmit Hazay and Yehuda Lindell. "A Note on the Relation between the Definitions of Security for Semi-Honest and Malicious Adversaries". In: *IACR Cryptol. ePrint Arch.* (2010), p. 551. URL: http://eprint.iacr.org/2010/551.

[HL10b] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. Information Security and Cryptography. Springer, 2010. ISBN: 978-3-642-14302-1. DOI: 10.1007/978-3-642-14303-8. URL: https://doi.org/10.1007/978-3-642-14303-8.

参考文献 IX

[Hon+19] Cheng Hong et al. “Covert Security with Public Verifiability: Faster, Leaner, and Simpler”. In: *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*. 2019, pp. 97–121. DOI: 10.1007/978-3-030-17659-4_4. URL: https://doi.org/10.1007/978-3-030-17659-4_4.

参考文献 X

[KM15] Vladimir Kolesnikov and Alex J. Malozemoff. "Public Verifiability in the Covert Model (Almost) for Free". In: *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*. 2015, pp. 210–235. DOI: 10.1007/978-3-662-48800-3_9. URL: https://doi.org/10.1007/978-3-662-48800-3_9.

[KLR10] Eyal Kushilevitz, Yehuda Lindell, and Tal Rabin. "Information-Theoretically Secure Protocols and Security under Composition". In: *SIAM J. Comput.* 39.5 (2010), pp. 2090–2112. DOI: 10.1137/090755886. URL: https://doi.org/10.1137/090755886.

参考文献 XI

[Lin09] Andrew Y. Lindell. "Adaptively Secure Two-Party Computation with Erasures". In: *Topics in Cryptology - CT-RSA 2009, The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, April 20-24, 2009. Proceedings*. 2009, pp. 117–132. DOI: 10.1007/978-3-642-00862-7_8. URL: https://doi.org/10.1007/978-3-642-00862-7_8.

[Lin16] Yehuda Lindell. "How To Simulate It - A Tutorial on the Simulation Proof Technique". In: *IACR Cryptol. ePrint Arch.* (2016), p. 46. URL: http://eprint.iacr.org/2016/046.

[Lin20] Yehuda Lindell. *Secure Multiparty Computation (MPC)*. Cryptology ePrint Archive, Paper 2020/300. https://eprint.iacr.org/2020/300. 2020. DOI: 10.1145/3387108. URL: https://eprint.iacr.org/2020/300.

参考文献 XII

[LP11] Yehuda Lindell and Benny Pinkas. “Secure Two-Party Computation via Cut-and-Choose Oblivious Transfer”. In: *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*. 2011, pp. 329–346. DOI: 10.1007/978-3-642-19571-6_20. URL: https://doi.org/10.1007/978-3-642-19571-6_20.

[OY91] Rafail Ostrovsky and Moti Yung. “How to Withstand Mobile Virus Attacks (Extended Abstract)”. In: *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing, Montreal, Quebec, Canada, August 19-21, 1991*. 1991, pp. 51–59. DOI: 10.1145/112600.112605. URL: https://doi.org/10.1145/112600.112605.

参考文献 XIII

[SSS21] Peter Scholl, Mark Simkin, and Luisa Siniscalchi. "Multiparty Computation with Covert Security and Public Verifiability". In: *IACR Cryptol. ePrint Arch.* (2021), p. 366. URL: https://eprint.iacr.org/2021/366.

[Yao86] Andrew Chi-Chih Yao. "How to Generate and Exchange Secrets (Extended Abstract)". In: *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*. 1986, pp. 162–167. DOI: 10.1109/SFCS.1986.25. URL: https://doi.org/10.1109/SFCS.1986.25.

参考文献 XIV

[Yao82] Andrew Chi-Chih Yao. “Protocols for Secure Computations (Extended Abstract)”. In: *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*. IEEE Computer Society, 1982, pp. 160–164. DOI: 10.1109/SFCS.1982.38. URL: https://doi.org/10.1109/SFCS.1982.38.

Thanks!