安全模型

《安全多方计算——可证明安全视角》第四章

2025 年 12 月 26 日

目录

什么是隐私性？

允许值

被攻陷方自己的输入和输出：

- $\{x_j, y_j\}_{P_j \in C}$

泄露值

被攻陷方在协议运行过程中看到的所有信息：

- $\{\text{view}_j\}_{P_j \in C}$

什么是隐私性？

允许值

被攻陷方自己的输入和输出：

- $\{x_j, y_j\}_{P_j \in C}$

泄露值

被攻陷方在协议运行过程中看到的所有信息：

- $\{\mathsf{view}_j\}_{P_j \in C}$

如果一个协议总是满足泄露值不包含比允许值更多的信息，那么它具有隐私性.

什么是隐私性？

允许值

被攻陷方自己的输入和输出：

- $\{x_j, y_j\}_{P_j \in C}$

泄露值

被攻陷方在协议运行过程中看到的所有信息：

- $\{\text{view}_j\}_{P_j \in C}$

如果一个协议总是满足泄露值不包含比允许值更多的信息，那么它具有隐私性：

- Q：如何定义一个值不包含比另一个值更多的信息呢？

什么是隐私性？

允许值

被攻陷方自己的输入和输出：

- $\{x_j, y_j\}_{P_j \in C}$

泄露值

被攻陷方在协议运行过程中看到的所有信息：

- $\{\mathsf{view}_j\}_{P_j \in C}$

如果一个协议总是满足泄露值不包含比允许值更多的信息，那么它具有隐私性.

- Q：如何定义一个值不包含比另一个值更多的信息呢？
- A：如果 V_1 可以从 V_2 高效地计算出来，我们就说 V_1 不包含比 V_2 更多的信息.

什么是隐私性？

允许值

被攻陷方自己的输入和输出：

- $\{x_j, y_j\}_{P_j \in C}$

泄露值

被攻陷方在协议运行过程中看到的所有信息：

- $\{\mathsf{view}_j\}_{P_j \in C}$

如果一个协议总是满足泄露值不包含比允许值更多的信息，那么它具有隐私性.

- Q：如何定义一个值不包含比另一个值更多的信息呢？
- A：如果 V_1 可以从 V_2 高效地计算出来，我们就说 V_1 不包含比 V_2 更多的信息.

$\Rightarrow$ 如果一个协议总是满足泄露值可以从允许值高效地计算出来，那么它具有隐私性.

什么是隐私性？

允许值

被攻陷方自己的输入和输出：

- $\{x_j, y_j\}_{P_j \in C}$

泄露值

被攻陷方在协议运行过程中看到的所有信息：

- $\{\mathsf{view}_j\}_{P_j \in C}$

如果一个协议总是满足泄露值不包含比允许值更多的信息，那么它具有隐私性.

- Q：如何定义一个值不包含比另一个值更多的信息呢？
- A：如果 V_1 可以从 V_2 高效地计算出来，我们就说 V_1 不包含比 V_2 更多的信息.

$\Rightarrow$ 如果一个协议总是满足泄露值可以从允许值高效地计算出来，那么它具有隐私性.

注意：如果参与方是随机算法，允许值和泄露值都是随机变量！所以……

什么是隐私性？

允许值

被攻陷方自己的输入和输出：

- $\{x_j, y_j\}_{P_j \in C}$

泄露值

被攻陷方在协议运行过程中看到的所有信息：

- $\{\mathsf{view}_j\}_{P_j \in C}$

如果一个协议总是满足泄露值不包含比允许值更多的信息，那么它具有隐私性.

- Q：如何定义一个值不包含比另一个值更多的信息呢？
- A：如果 V_1 可以从 V_2 高效地计算出来，我们就说 V_1 不包含比 V_2 更多的信息.

$\Rightarrow$ 如果一个协议总是满足泄露值可以从允许值高效地计算出来，那么它具有隐私性.

注意：如果参与方是随机算法，允许值和泄露值都是随机变量！所以……

隐私性

如果存在一个高效的模拟器 $\mathcal{S}$，满足模拟值 $\mathcal{S}(\{x_j, y_j\}_{P_j \in C})$ 和泄露值 $\{\mathsf{view}_j\}_{P_j \in C}$ 具有不可区分的分布，那么我们说这个协议具有隐私性.

什么是隐私性？

允许值

被攻陷方自己的输入和输出：

- $\{x_j, y_j\}_{P_j \in C}$

泄露值

被攻陷方在协议运行过程中看到的所有信息：

- $\{\mathsf{view}_j\}_{P_j \in C}$

如果一个协议总是满足泄露值不包含比允许值更多的信息，那么它具有隐私性.

- Q：如何定义一个值不包含比另一个值更多的信息呢？
- A：如果 V_1 可以从 V_2 高效地计算出来，我们就说 V_1 不包含比 V_2 更多的信息.

$\Rightarrow$ 如果一个协议总是满足泄露值可以从允许值高效地计算出来，那么它具有隐私性.

注意：如果参与方是随机算法，允许值和泄露值都是随机变量！所以……

隐私性

如果存在一个高效的模拟器 $\mathcal{S}$，满足模拟值 $\mathcal{S}(\{x_j, y_j\}_{P_j \in C})$ 和泄露值 $\{\mathsf{view}_j\}_{P_j \in C}$ 具有不可区分的分布，那么我们说这个协议具有隐私性.

还没完……

“愚蠢”功能及其具体实现

理想功能 $\mathcal{F}_{fool}$

没有任何输入. P_1 输出一个随机比特; P_2 没有输出.

“愚蠢”功能及其具体实现

理想功能 $\mathcal{F}_{fool}$

没有任何输入. P_1 输出一个随机比特; P_2 没有输出.

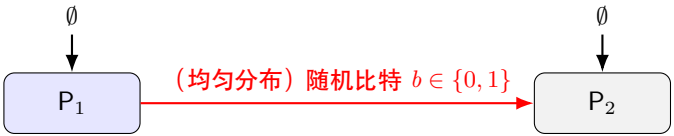
- 协议: P_1 没有输入，取随机比特并输出，同时把该比特发送给 P_2，P_2 没有输入输出.

“愚蠢”功能及其具体实现

理想功能 $\mathcal{F}_{fool}$

没有任何输入. P_1 输出一个随机比特; P_2 没有输出.

- 协议: P_1 没有输入，取随机比特并输出，同时把该比特发送给 P_2，P_2 没有输入输出.

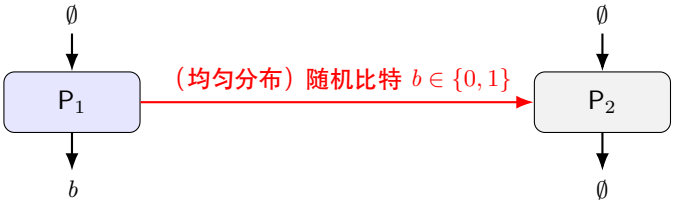


“愚蠢”功能及其具体实现

理想功能 $\mathcal{F}_{fool}$

没有任何输入. P_1 输出一个随机比特; P_2 没有输出.

- 协议: P_1 没有输入，取随机比特并输出，同时把该比特发送给 P_2，P_2 没有输入输出.

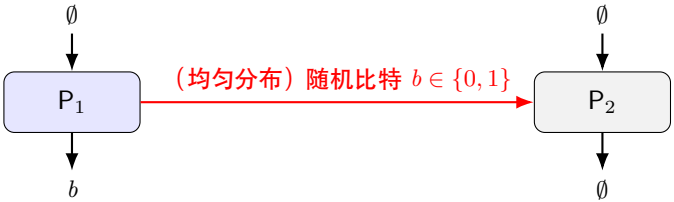


“愚蠢”功能及其具体实现

理想功能 $\mathcal{F}_{fool}$

没有任何输入. P_1 输出一个随机比特; P_2 没有输出.

- 协议: P_1 没有输入，取随机比特并输出，同时把该比特发送给 P_2，P_2 没有输入输出.



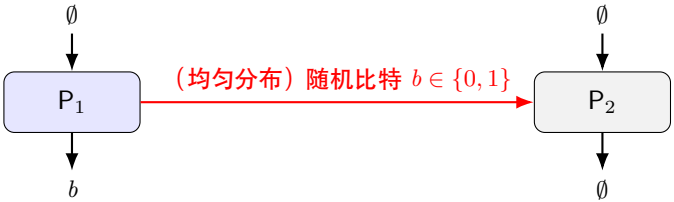
- Q: 这个协议安全吗？

“愚蠢”功能及其具体实现

理想功能 $\mathcal{F}_{fool}$

没有任何输入. P_1 输出一个随机比特; P_2 没有输出.

- 协议: P_1 没有输入，取随机比特并输出，同时把该比特发送给 P_2，P_2 没有输入输出.



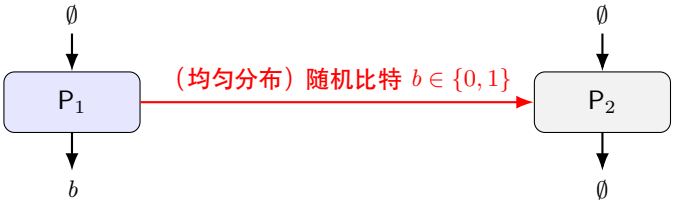
- Q: 这个协议安全吗？
- A: 不安全！因为 P_2 能够学到 b，而理想功能并没有泄露给 P_2 任何信息.

“愚蠢”功能及其具体实现

理想功能 $\mathcal{F}_{fool}$

没有任何输入. P_1 输出一个随机比特; P_2 没有输出.

- 协议: P_1 没有输入，取随机比特并输出，同时把该比特发送给 P_2，P_2 没有输入输出.



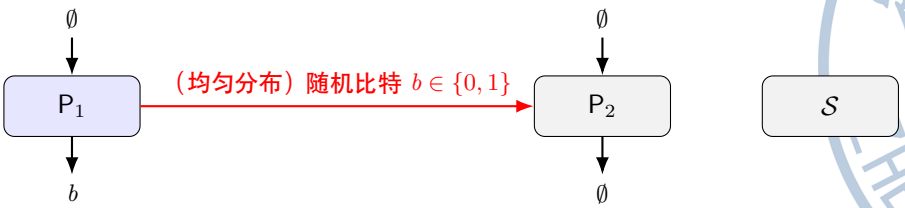
- Q: 这个协议安全吗？
- A: 不安全！因为 P_2 能够学到 b，而理想功能并没有泄露给 P_2 任何信息.
- Q: 但泄露的比特好像是可以模拟的呀……

“愚蠢”功能及其具体实现

理想功能 $\mathcal{F}_{fool}$

没有任何输入. P_1 输出一个随机比特; P_2 没有输出.

- 协议: P_1 没有输入，取随机比特并输出，同时把该比特发送给 P_2，P_2 没有输入输出.



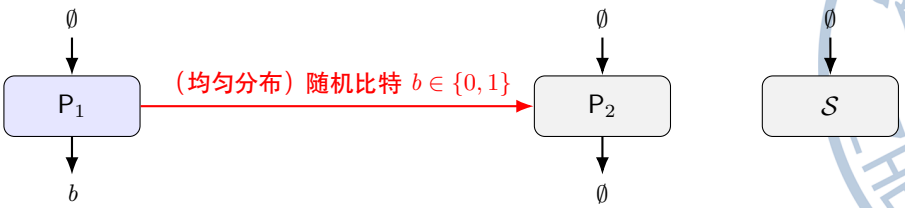
- Q: 这个协议安全吗？
- A: 不安全！因为 P_2 能够学到 b，而理想功能并没有泄露给 P_2 任何信息.
- Q: 但泄露的比特好像是可以模拟的呀……

“愚蠢”功能及其具体实现

理想功能 $\mathcal{F}_{fool}$

没有任何输入. P_1 输出一个随机比特; P_2 没有输出.

- 协议: P_1 没有输入，取随机比特并输出，同时把该比特发送给 P_2，P_2 没有输入输出.



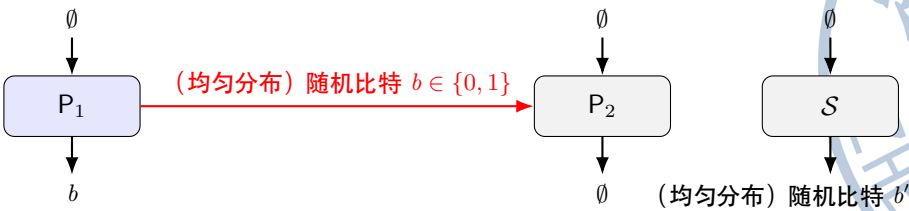
- Q: 这个协议安全吗？
- A: 不安全！因为 P_2 能够学到 b，而理想功能并没有泄露给 P_2 任何信息.
- Q: 但泄露的比特好像是可以模拟的呀……

“愚蠢”功能及其具体实现

理想功能 $\mathcal{F}_{fool}$

没有任何输入. P_1 输出一个随机比特; P_2 没有输出.

- 协议: P_1 没有输入，取随机比特并输出，同时把该比特发送给 P_2，P_2 没有输入输出.



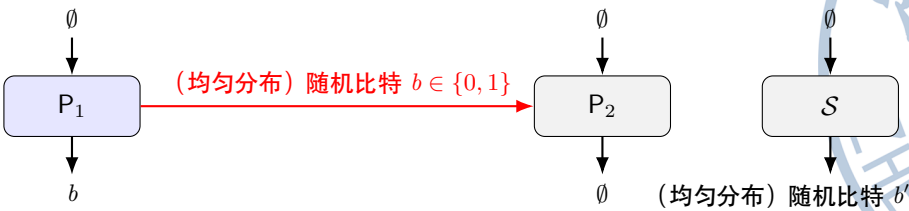
- Q: 这个协议安全吗？
- A: 不安全！因为 P_2 能够学到 b，而理想功能并没有泄露给 P_2 任何信息.
- Q: 但泄露的比特好像是可以模拟的呀……

“愚蠢”功能及其具体实现

理想功能 $\mathcal{F}_{fool}$

没有任何输入. P_1 输出一个随机比特; P_2 没有输出.

- 协议: P_1 没有输入，取随机比特并输出，同时把该比特发送给 P_2，P_2 没有输入输出.



- Q: 这个协议安全吗？
- A: 不安全！因为 P_2 能够学到 b，而理想功能并没有泄露给 P_2 任何信息.
- Q: 但泄露的比特好像是可以模拟的呀……b 和 b' 都是均匀随机分布的呀？

模拟的细节：联合分布

- 前面的模拟没有体现模拟值/泄露值与诚实方输出的关系.

模拟的细节：联合分布

- 前面的模拟没有体现模拟值/泄露值与诚实方输出的关系.
  - 泄露值：P_2 的视图 $\text{view}_2 = \{b\}$，与 P_1 的输出相同.

模拟的细节：联合分布

- 前面的模拟没有体现模拟值/泄露值与诚实方输出的关系.
  - 泄露值：P_2 的视图 $\text{view}_2 = \{b\}$，与 P_1 的输出相同.
  - 模拟值：$\mathcal{S}(\emptyset) = \{b'\}$，与 P_1 的输出无关.

模拟的细节：联合分布

- 前面的模拟没有体现模拟值/泄露值与诚实方输出的关系.
  - 泄露值：P_2 的视图 $\text{view}_2 = \{b\}$，与 P_1 的输出相同.
  - 模拟值：$\mathcal{S}(\emptyset) = \{b'\}$，与 P_1 的输出无关.
- **我们必须考虑模拟值/泄露值与诚实方输出的联合分布!**

模拟的细节：联合分布

- 前面的模拟没有体现模拟值/泄露值与诚实方输出的关系.
  - 泄露值：P_2 的视图 $\text{view}_2 = \{b\}$，与 P_1 的输出相同.
  - 模拟值：$\mathcal{S}(\emptyset) = \{b'\}$，与 P_1 的输出无关.
- **我们必须考虑模拟值/泄露值与诚实方输出的联合分布!**

泄露值与诚实方输出的联合分布

	$\text{Output}_1\ b=0$	$\text{Output}_1\ b=1$
$\text{View}_2\ b=0$	$1/2$	0
$\text{View}_2\ b=1$	0	$1/2$

模拟值与诚实方输出的联合分布

	$\text{Output}_1\ b=0$	$\text{Output}_1\ b=1$
$\text{Sim}_2\ b'=0$	$1/4$	$1/4$
$\text{Sim}_2\ b'=1$	$1/4$	$1/4$

模拟的细节：联合分布

- 前面的模拟没有体现模拟值/泄露值与诚实方输出的关系.
  - 泄露值：P_2 的视图 $\text{view}_2 = \{b\}$，与 P_1 的输出相同.
  - 模拟值：$\mathcal{S}(\emptyset) = \{b'\}$，与 P_1 的输出无关.
- **我们必须考虑模拟值/泄露值与诚实方输出的联合分布!**

泄露值与诚实方输出的联合分布

	$\text{Output}_1\ b=0$	$\text{Output}_1\ b=1$
$\text{View}_2\ b=0$	$1/2$	0
$\text{View}_2\ b=1$	0	$1/2$

模拟值与诚实方输出的联合分布

	$\text{Output}_1\ b=0$	$\text{Output}_1\ b=1$
$\text{Sim}_2\ b'=0$	$1/4$	$1/4$
$\text{Sim}_2\ b'=1$	$1/4$	$1/4$

- 由于联合分布不同，模拟失败!

模拟的细节：联合分布

- 前面的模拟没有体现模拟值/泄露值与诚实方输出的关系.
  - 泄露值：P_2 的视图 $\text{view}_2 = \{b\}$，与 P_1 的输出相同.
  - 模拟值：$\mathcal{S}(\emptyset) = \{b'\}$，与 P_1 的输出无关.
- **我们必须考虑模拟值/泄露值与诚实方输出的联合分布!**

泄露值与诚实方输出的联合分布

	$\text{Output}_1\ b=0$	$\text{Output}_1\ b=1$
$\text{View}_2\ b=0$	$1/2$	0
$\text{View}_2\ b=1$	0	$1/2$

模拟值与诚实方输出的联合分布

	$\text{Output}_1\ b=0$	$\text{Output}_1\ b=1$
$\text{Sim}_2\ b'=0$	$1/4$	$1/4$
$\text{Sim}_2\ b'=1$	$1/4$	$1/4$

- 由于联合分布不同，模拟失败!
- P_2 的视图与 P_1 的输出是完全相关的，而**模拟器没有诚实方 P_1 的输出信息**，因此无法模拟这种相关性，符合协议不安全的直觉.

模拟的细节：联合分布

- 前面的模拟没有体现模拟值/泄露值与诚实方输出的关系.
  - 泄露值：P_2 的视图 $\text{view}_2 = \{b\}$，与 P_1 的输出相同.
  - 模拟值：$\mathcal{S}(\emptyset) = \{b'\}$，与 P_1 的输出无关.
- **我们必须考虑模拟值/泄露值与诚实方输出的联合分布!**

泄露值与诚实方输出的联合分布

	$\text{Output}_1\ b=0$	$\text{Output}_1\ b=1$
$\text{View}_2\ b=0$	$1/2$	0
$\text{View}_2\ b=1$	0	$1/2$

模拟值与诚实方输出的联合分布

	$\text{Output}_1\ b=0$	$\text{Output}_1\ b=1$
$\text{Sim}_2\ b'=0$	$1/4$	$1/4$
$\text{Sim}_2\ b'=1$	$1/4$	$1/4$

- 由于联合分布不同，模拟失败!
- P_2 的视图与 P_1 的输出是完全相关的，而**模拟器没有诚实方 P_1 的输出信息**，因此无法模拟这种相关性，符合协议不安全的直觉.

注：如果理想功能是确定性的（没有随机性），则联合分布退化为只需考虑模拟值/泄露值的分布.

什么是恶意安全性？

半诚实模型

- 严格遵守协议步骤.
- 但会记录视图，试图推断隐私.

恶意模型

- 可以任意偏离协议.
- 目标：破坏正确性、获取隐私等.

什么是恶意安全性？

半诚实模型

- 严格遵守协议步骤.
- 但会记录视图，试图推断隐私.

恶意模型

- 可以任意偏离协议.
- 目标：破坏正确性、获取隐私等.

恶意敌手关心的不只是**获得信息**，而是**产生影响**：

什么是恶意安全性？

半诚实模型

- 严格遵守协议步骤.
- 但会记录视图，试图推断隐私.

恶意模型

- 可以任意偏离协议.
- 目标：破坏正确性、获取隐私等.

恶意敌手关心的不只是**获得信息**，而是**产生影响**：

- 可以随意偏离协议、发送任意消息.
- 可能改变协议输出或让协议中止.

什么是恶意安全性？

半诚实模型

- 严格遵守协议步骤.
- 但会记录视图，试图推断隐私.

恶意模型

- 可以任意偏离协议.
- 目标：破坏正确性、获取隐私等.

恶意敌手关心的不只是**获得信息**，而是**产生影响**：

- 可以随意偏离协议、发送任意消息.
- 可能改变协议输出或让协议中止.

Q：什么样的影响会破坏协议的安全性？

什么是恶意安全性？

半诚实模型

- 严格遵守协议步骤.
- 但会记录视图，试图推断隐私.

恶意模型

- 可以任意偏离协议.
- 目标：破坏正确性、获取隐私等.

恶意敌手关心的不只是**获得信息**，而是**产生影响**：

- 可以随意偏离协议、发送任意消息.
- 可能改变协议输出或让协议中止.

Q：什么样的影响会破坏协议的安全性？
A：那些无法通过理想功能中的攻击实现的影响.

例 4.1：输入替换

理想功能与门限设定

三方计算 $f(x_1, x_2, x_3) = (x_1x_2x_3, x_2, x_3)$，$t = 1$.

例 4.1：输入替换

理想功能与门限设定

三方计算 $f(x_1, x_2, x_3) = (x_1x_2x_3, x_2, x_3)$，$t = 1$.

- **攻击**：P_1 **被攻陷，将输入替换为** $x_1' = 1$.

例 4.1：输入替换

理想功能与门限设定

三方计算 $f(x_1, x_2, x_3) = (x_1x_2x_3, x_2, x_3)$，$t = 1$.

- **攻击**：P_1 **被攻陷，将输入替换为** $x_1' = 1$.
- **结果**：P_1 **得到** $y_1 = x_1'x_2x_3 = x_2x_3$.

例 4.1：输入替换

理想功能与门限设定

三方计算 $f(x_1, x_2, x_3) = (x_1x_2x_3, x_2, x_3)$，$t = 1$.

- 攻击：P_1 被攻陷，将输入替换为 $x_1' = 1$.
- 结果：P_1 得到 $y_1 = x_1'x_2x_3 = x_2x_3$.

结论

这种攻击是对理想功能的攻击，等价于诚实地用输入 x_1' 运行协议.
造成的影响无法通过协议的设计来防止，是**允许的影响**.

例 4.2：篡改中间值

理想功能与门限设定

三方计算 $f(x_1, x_2, x_3) = (x_1x_2x_3, x_2, x_3)$，$t = 1$.

例 4.2：篡改中间值

理想功能与门限设定

三方计算 $f(x_1, x_2, x_3) = (x_1x_2x_3, x_2, x_3)$，$t = 1$.

按照这样的顺序安排电路中的门：

1. 计算 $[u]_t = [x_1x_2]_t$；
2. 计算 $[y_1]_t = [ux_3]_t$；
3. 输出阶段重建 y_1.

- 假设 P_1 被攻陷. P_1 在输入共享阶段使用输入 $x_1' = 0$ 并遵循协议.

例 4.2：篡改中间值

理想功能与门限设定

三方计算 $f(x_1, x_2, x_3) = (x_1x_2x_3, x_2, x_3)$，$t = 1$.

按照这样的顺序安排电路中的门：

1. 计算 $[u]_t = [x_1x_2]_t$；
2. 计算 $[y_1]_t = [ux_3]_t$；
3. 输出阶段重建 y_1.

- 假设 P_1 被攻陷．P_1 在输入共享阶段使用输入 $x_1' = 0$ 并遵循协议.
- 然而，在执行乘法协议计算 $[u]_t = [x_1'x_2]_t$ 时，各参与方在本地计算多项式 h，其中 $h(0) = x_1'x_2$.

例 4.2：篡改中间值

理想功能与门限设定

三方计算 $f(x_1, x_2, x_3) = (x_1x_2x_3, x_2, x_3)$，$t = 1$.

按照这样的顺序安排电路中的门：

1. 计算 $[u]_t = [x_1x_2]_t$；
2. 计算 $[y_1]_t = [ux_3]_t$；
3. 输出阶段重建 y_1.

- 假设 P_1 被攻陷．P_1 在输入共享阶段使用输入 $x_1' = 0$ 并遵循协议.
- 然而，在执行乘法协议计算 $[u]_t = [x_1'x_2]_t$ 时，各参与方在本地计算多项式 h，其中 $h(0) = x_1'x_2$.
- P_1 没有按照协议规定的方式分发 $[h(\alpha_1)]_t$，而是分发了 $[r_1^{-1} + h(\alpha_1)]_t$，$r_1$ 是重组向量中的第一个元素.

例 4.2：篡改中间值

这意味着结果变成了：

$$
\begin{aligned}
[u]_t &= r_1[r_1^{-1} + h(\alpha_1)]_t + \sum_{i=2}^{3} r_i[h(\alpha_i)]_t \\
&= [r_1 r_1^{-1} + r_1 h(\alpha_1) + \sum_{i=2}^{3} r_i h(\alpha_i)]_t \\
&= [1 + \sum_{i=1}^{3} r_i h(\alpha_i)]_t \\
&= [1 + x_1' x_2]_t \\
&= [1]_t
\end{aligned}
$$

- 由于 $x_1' = 0$，所以最后一个等号成立.

例 4.2：篡改中间值

这意味着结果变成了：

$$
\begin{aligned}
[u]_t &= r_1[r_1^{-1} + h(\alpha_1)]_t + \sum_{i=2}^{3} r_i[h(\alpha_i)]_t \\
&= [r_1 r_1^{-1} + r_1 h(\alpha_1) + \sum_{i=2}^{3} r_i h(\alpha_i)]_t \\
&= [1 + \sum_{i=1}^{3} r_i h(\alpha_i)]_t \\
&= [1 + x_1' x_2]_t \\
&= [1]_t
\end{aligned}
$$

- 由于 $x_1' = 0$，所以最后一个等号成立.
- P_1 在乘法协议中诚实地计算 $[y_1]_t = [ux_3]_t$，并遵循输出重建阶段的协议.

例 4.2：篡改中间值

这意味着结果变成了：

$$
\begin{aligned}
[u]_t &= r_1[r_1^{-1} + h(\alpha_1)]_t + \sum_{i=2}^{3} r_i[h(\alpha_i)]_t \\
&= [r_1 r_1^{-1} + r_1 h(\alpha_1) + \sum_{i=2}^{3} r_i h(\alpha_i)]_t \\
&= [1 + \sum_{i=1}^{3} r_i h(\alpha_i)]_t \\
&= [1 + x_1' x_2]_t \\
&= [1]_t
\end{aligned}
$$

- 由于 $x_1' = 0$，所以最后一个等号成立.
- P_1 在乘法协议中诚实地计算 $[y_1]_t = [ux_3]_t$，并遵循输出重建阶段的协议.
- 最终 P_1 将得到 $y_1 = ux_3 = 1 \cdot x_3 = x_3$.

例 4.2：篡改中间值

这意味着结果变成了：

$$
\begin{aligned}
[u]_t &= r_1[r_1^{-1} + h(\alpha_1)]_t + \sum_{i=2}^{3} r_i[h(\alpha_i)]_t \\
&= [r_1 r_1^{-1} + r_1 h(\alpha_1) + \sum_{i=2}^{3} r_i h(\alpha_i)]_t \\
&= [1 + \sum_{i=1}^{3} r_i h(\alpha_i)]_t \\
&= [1 + x_1' x_2]_t \\
&= [1]_t
\end{aligned}
$$

- 由于 $x_1' = 0$，所以最后一个等号成立.
- P_1 在乘法协议中诚实地计算 $[y_1]_t = [ux_3]_t$，并遵循输出重建阶段的协议.
- 最终 P_1 将得到 $y_1 = ux_3 = 1 \cdot x_3 = x_3$.

结论

这是**不允许的影响**：理想功能中 P_1 无法总是得到 x_3（如 $x_2 = 0$ 时）.

恶意安全性的定义

允许影响和实际影响

恶意安全性的定义

允许影响和实际影响

- 允许影响：在理想功能中可能产生的影响，也就是进行**输入替换**.

恶意安全性的定义

允许影响和实际影响

- 允许影响：在理想功能中可能产生的影响，也就是进行**输入替换**.
- 实际影响：具体协议实现中可能产生的影响，敌手可代表被攻陷方发送任意消息.

恶意安全性的定义

允许影响和实际影响

- 允许影响：在理想功能中可能产生的影响，也就是进行**输入替换**.
- 实际影响：具体协议实现中可能产生的影响，敌手可代表被攻陷方发送任意消息.

如果任意实际影响的效果都能通过允许的影响得到，协议就是恶意安全的.

恶意安全性的定义

允许影响和实际影响

- 允许影响：在理想功能中可能产生的影响，也就是进行**输入替换**.
- 实际影响：具体协议实现中可能产生的影响，敌手可代表被攻陷方发送任意消息.

如果任意实际影响的效果都能通过允许的影响得到，协议就是恶意安全的.

恶意安全性

如果对于每个攻击协议的敌手，都存在一个模拟器 $\mathcal{S}$ 可以高效地计算出具有相同效果的允许影响，那么该协议是恶意安全的.

完整的安全性 = 隐私性 + 恶意安全性

完整的安全性 = 隐私性 + 恶意安全性

- **隐私性**：被攻陷方只能学到允许值的信息.

完整的安全性 = 隐私性 + 恶意安全性

- **隐私性**：被攻陷方只能学到允许值的信息.
- **恶意安全性**：被攻陷方只能对输出产生允许的影响.

完整的安全性 = 隐私性 + 恶意安全性

- **隐私性**：被攻陷方只能学到允许值的信息.
- **恶意安全性**：被攻陷方只能对输出产生允许的影响.

完整的安全性

存在一个单一的模拟器，同时保证隐私性和恶意安全性：

- 模拟器接收敌手试图影响真实协议的行为，并高效地将其转化为允许的影响 (输入替换)；
- 模拟器接收允许值，并高效地向敌手模拟泄露值.

完整的安全性 = 隐私性 + 恶意安全性

- **隐私性**：被攻陷方只能学到允许值的信息.
- **恶意安全性**：被攻陷方只能对输出产生允许的影响.

完整的安全性

存在一个单一的模拟器，同时保证隐私性和恶意安全性：

- 模拟器接收敌手试图影响真实协议的行为，并高效地将其转化为允许的影响 (输入替换)；
- 模拟器接收允许值，并高效地向敌手模拟泄露值.

Q：如何给出形式化定义？

完整的安全性 = 隐私性 + 恶意安全性

- **隐私性**：被攻陷方只能学到允许值的信息.
- **恶意安全性**：被攻陷方只能对输出产生允许的影响.

完整的安全性

存在一个单一的模拟器，同时保证隐私性和恶意安全性：

- 模拟器接收敌手试图影响真实协议的行为，并高效地将其转化为允许的影响 (输入替换)；
- 模拟器接收允许值，并高效地向敌手模拟泄露值.

Q：如何给出形式化定义？
A：请听我用 UC 娓娓道来……

Why UC?

独立 (Stand-alone) 模型

- 仅保证单次运行安全.
- 无法抵抗并发攻击.

通用可组合 (Universal Composability, UC) 框架

- 引入 **环境** $\mathcal{Z}$，$\mathcal{Z}$ 可与敌手实时交互.
- **任意环境/任意协议组合下仍安全**.

通用可组合定理：等价替换

调用理想功能 $\mathcal{F}$ 实现某复杂协议

```
Protocol Pi_CMPLX^F:
  ...
  call F on inputs
  use F's outputs
  ...
```

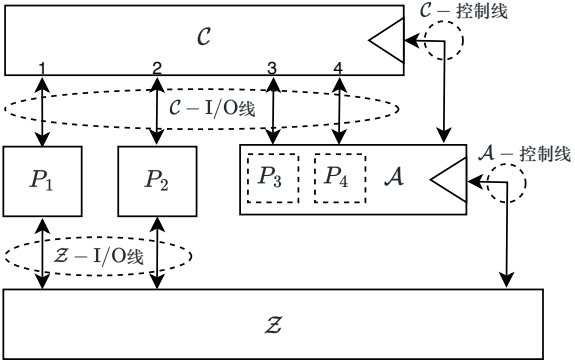
$\equiv$

调用安全协议 Π 实现某复杂协议

```
Protocol Pi_CMPLX^Pi:
  ...
  run Pi as subprotocol on inputs
  use Pi's outputs
  ...
```

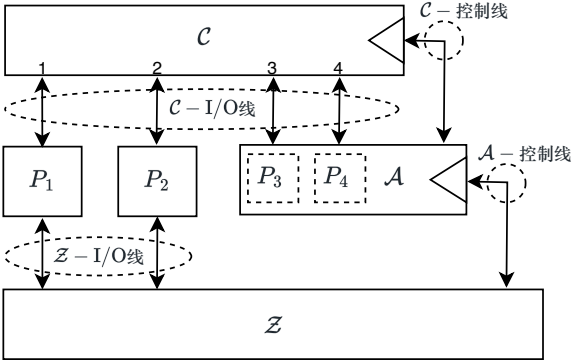
安全性等价：Π UC-安全实现 $\mathcal{F}$ 时，可安全替换 $\mathcal{F}$ 为 Π.

真实协议运行结构



- $\mathcal{C}$ 连接所有参与方；
- 被攻陷方接入 $\mathcal{A}$；
- $\mathcal{Z}$ 与诚实方交互并与敌手 $\mathcal{A}$ 通信；

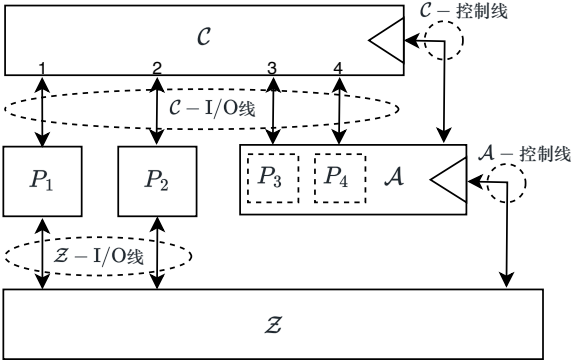
真实协议运行结构



- $\mathcal{C}$ 连接所有参与方；
- 被攻陷方接入 $\mathcal{A}$；
- $\mathcal{Z}$ 与诚实方交互并与敌手 $\mathcal{A}$ 通信；

- $\mathcal{Z} \to P_i$：输入（$\mathcal{Z}$-I/O）.

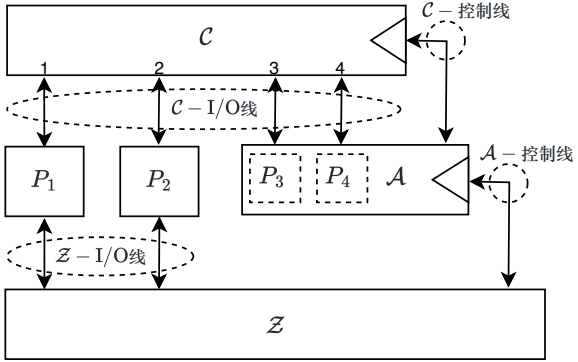
真实协议运行结构



- $\mathcal{C}$ 连接所有参与方；
- 被攻陷方接入 $\mathcal{A}$；
- $\mathcal{Z}$ 与诚实方交互并与敌手 $\mathcal{A}$ 通信；

- $\mathcal{Z} \to P_i$：输入（$\mathcal{Z}$-I/O）.
- $\mathcal{Z} \to \mathcal{A}$：传递控制消息.

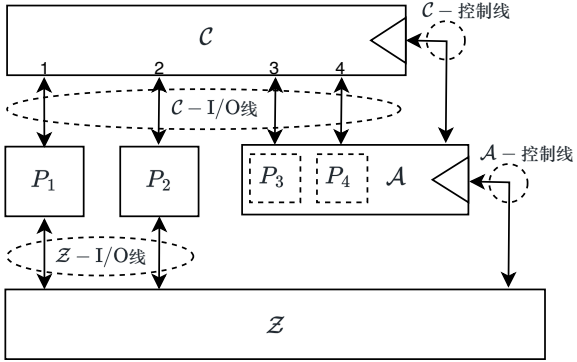
真实协议运行结构



- $\mathcal{C}$ 连接所有参与方；
- 被攻陷方接入 $\mathcal{A}$；
- $\mathcal{Z}$ 与诚实方交互并与敌手 $\mathcal{A}$ 通信；

- $\mathcal{Z} \to P_i$：输入（$\mathcal{Z}$-I/O）.
- $\mathcal{Z} \to \mathcal{A}$：传递控制消息.
- $P_i/\mathcal{A} \to \mathcal{C}$：发送请求（$\textsc{Send}, j, m$）.

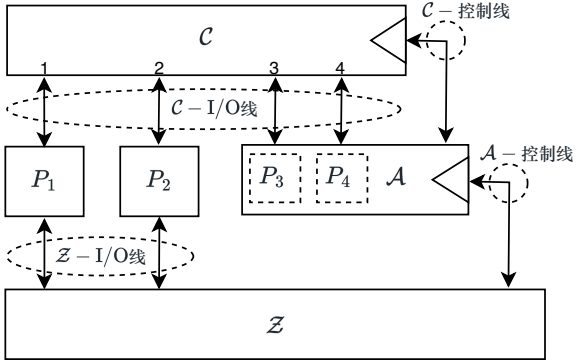
真实协议运行结构



- $\mathcal{C}$ 连接所有参与方；
- 被攻陷方接入 $\mathcal{A}$；
- $\mathcal{Z}$ 与诚实方交互并与敌手 $\mathcal{A}$ 通信；

- $\mathcal{Z} \to P_i$：输入（$\mathcal{Z}$-I/O）.
- $\mathcal{Z} \to \mathcal{A}$：传递控制消息.
- $P_i/\mathcal{A} \to \mathcal{C}$：发送请求（$\textsc{Send}, j, m$）.
- $\mathcal{C} \to \mathcal{A}$：泄露（$\textsc{Sent}, i, \mathrm{len}(m)$）.

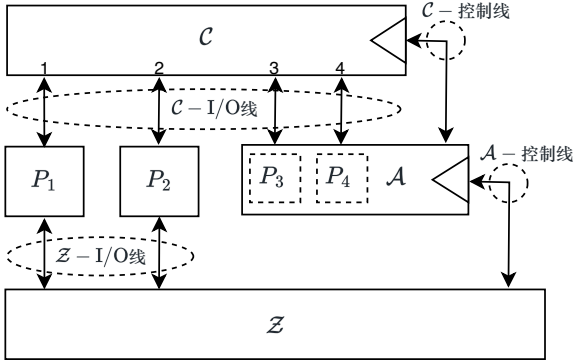
真实协议运行结构



- $\mathcal{C}$ 连接所有参与方；
- 被攻陷方接入 $\mathcal{A}$；
- $\mathcal{Z}$ 与诚实方交互并与敌手 $\mathcal{A}$ 通信；

- $\mathcal{Z} \to P_i$：输入（$\mathcal{Z}$-I/O）.
- $\mathcal{Z} \to \mathcal{A}$：传递控制消息.
- $P_i/\mathcal{A} \to \mathcal{C}$：发送请求（$\textsc{Send}, j, m$）.
- $\mathcal{C} \to \mathcal{A}$：泄露（$\textsc{Sent}, i, \mathrm{len}(m)$）.
- $\mathcal{A} \to \mathcal{Z}$：传递泄露消息；

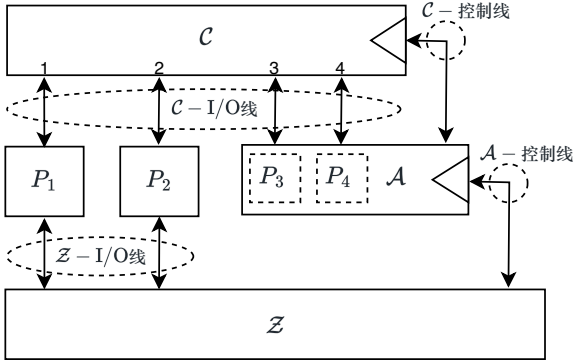
真实协议运行结构



- $\mathcal{C}$ 连接所有参与方；
- 被攻陷方接入 $\mathcal{A}$；
- $\mathcal{Z}$ 与诚实方交互并与敌手 $\mathcal{A}$ 通信；

- $\mathcal{Z} \to P_i$：输入（$\mathcal{Z}$-I/O）.
- $\mathcal{Z} \to \mathcal{A}$：传递控制消息.
- $P_i/\mathcal{A} \to \mathcal{C}$：发送请求（$\textsc{Send}, j, m$）.
- $\mathcal{C} \to \mathcal{A}$：泄露（$\textsc{Sent}, i, \text{len}(m)$）.
- $\mathcal{A} \to \mathcal{Z}$：传递泄露消息；
- $\mathcal{A} \to \mathcal{C}$：调度（$\textsc{Deliver}, k$）.

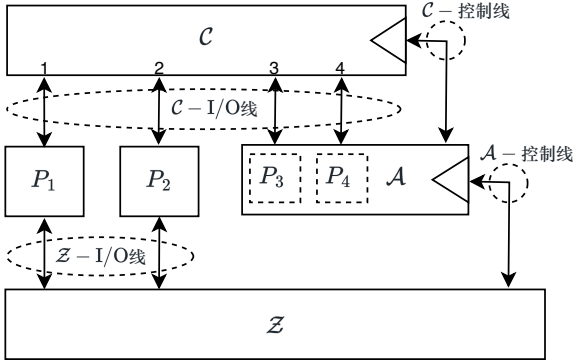
真实协议运行结构



- $\mathcal{C}$ 连接所有参与方；
- 被攻陷方接入 $\mathcal{A}$；
- $\mathcal{Z}$ 与诚实方交互并与敌手 $\mathcal{A}$ 通信；

- $\mathcal{Z} \to P_i$：输入（$\mathcal{Z}$-I/O）.
- $\mathcal{Z} \to \mathcal{A}$：传递控制消息.
- $P_i/\mathcal{A} \to \mathcal{C}$：发送请求（$\textsc{Send}, j, m$）.
- $\mathcal{C} \to \mathcal{A}$：泄露（$\textsc{Sent}, i, \mathrm{len}(m)$）.
- $\mathcal{A} \to \mathcal{Z}$：传递泄露消息；
- $\mathcal{A} \to \mathcal{C}$：调度（$\textsc{Deliver}, k$）.
- $\mathcal{C} \to P_j/\mathcal{A}$：传递（$\textsc{Receive}, i, m$）.

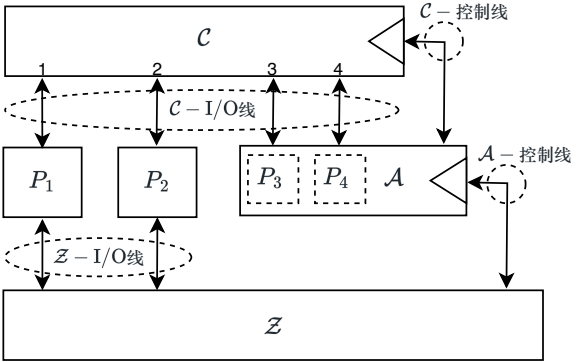
真实协议运行结构



- $\mathcal{C}$ 连接所有参与方；
- 被攻陷方接入 $\mathcal{A}$；
- $\mathcal{Z}$ 与诚实方交互并与敌手 $\mathcal{A}$ 通信；

- $\mathcal{Z} \to P_i$：输入（$\mathcal{Z}$-I/O）.
- $\mathcal{Z} \to \mathcal{A}$：传递控制消息.
- $P_i/\mathcal{A} \to \mathcal{C}$：发送请求（$\textsc{Send}, j, m$）.
- $\mathcal{C} \to \mathcal{A}$：泄露（$\textsc{Sent}, i, \mathrm{len}(m)$）.
- $\mathcal{A} \to \mathcal{Z}$：传递泄露消息；
- $\mathcal{A} \to \mathcal{C}$：调度（$\textsc{Deliver}, k$）.
- $\mathcal{C} \to P_j/\mathcal{A}$：传递（$\textsc{Receive}, i, m$）.
- $P_i \to \mathcal{Z}$：输出.

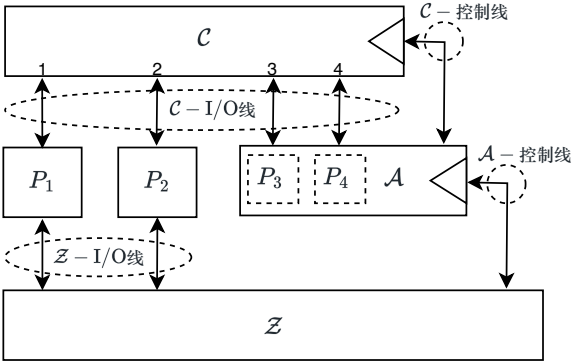
真实协议：敌手能做什么？



对通信的观察与控制

- 知道**谁**向**谁**发送、消息**长度**.
- 决定消息**是否/何时**被传递（调度顺序）.
- 但不能读取内容、不能篡改或改写发送方.

真实协议：敌手能做什么？



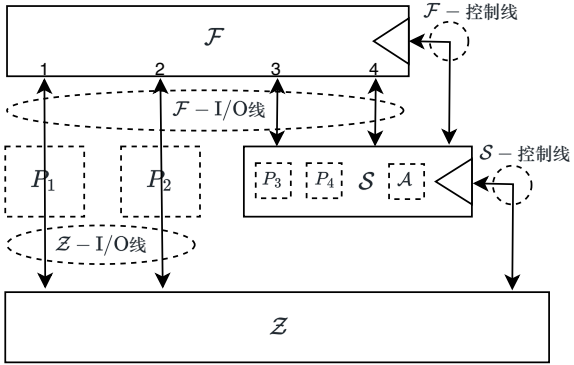
对通信的观察与控制

- 知道**谁**向**谁**发送、消息**长度**.
- 决定消息**是否/何时**被传递（调度顺序）.
- 但不能读取内容、不能篡改或改写发送方.

对参与方的控制

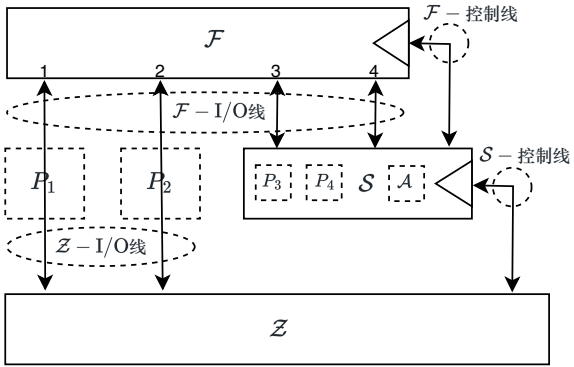
被攻陷方的全部行为由敌手 $\mathcal{A}$ 决定.

理想协议运行结构



- $\mathcal{F}$ 替代网络 $\mathcal{C}$.
- 被攻陷方接入模拟器 $\mathcal{S}$.
- $\mathcal{Z}$ 与“诚实方”交互，并与“敌手” $\mathcal{S}$ 通信.

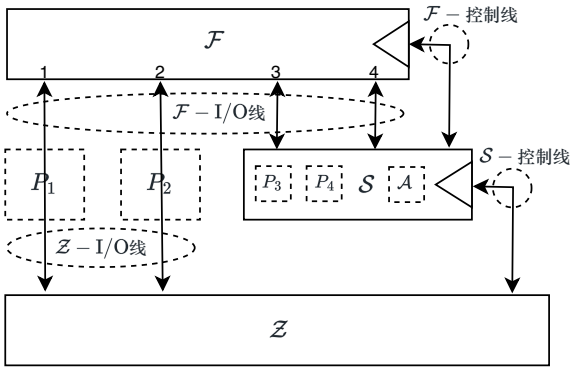
理想协议运行结构



- $\mathcal{F}$ 替代网络 $\mathcal{C}$.
- 被攻陷方接入模拟器 $\mathcal{S}$.
- $\mathcal{Z}$ 与“诚实方”交互，并与“敌手” $\mathcal{S}$ 通信.

- $\mathcal{F}$-I/O 线：连接诚实方与 $\mathcal{F}$（输入/输出转发）.

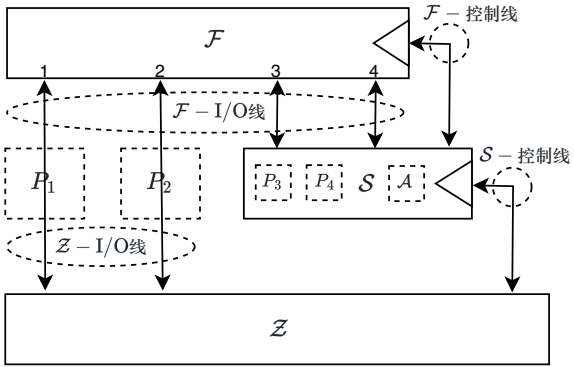
理想协议运行结构



- $\mathcal{F}$ 替代网络 $\mathcal{C}$.
- 被攻陷方接入模拟器 $\mathcal{S}$.
- $\mathcal{Z}$ 与“诚实方”交互，并与“敌手” $\mathcal{S}$ 通信.

- $\mathcal{F}$-I/O 线：连接诚实方与 $\mathcal{F}$（输入/输出转发）.
- $\mathcal{F}$-控制线：连接 $\mathcal{S}$ 与 $\mathcal{F}$（允许影响）.

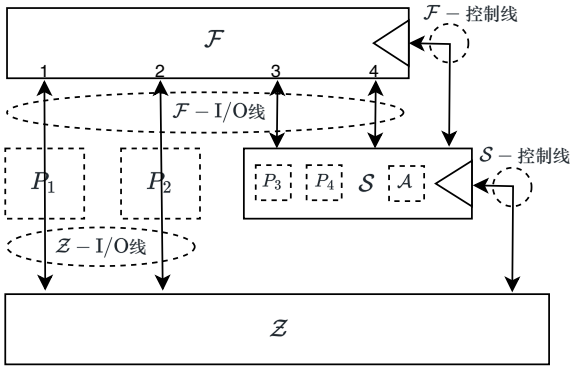
理想协议运行结构



- $\mathcal{F}$ 替代网络 $\mathcal{C}$.
- 被攻陷方接入模拟器 $\mathcal{S}$.
- $\mathcal{Z}$ 与“诚实方”交互，并与“敌手” $\mathcal{S}$ 通信.

- $\mathcal{F}$-I/O 线：连接诚实方与 $\mathcal{F}$（输入/输出转发）.
- $\mathcal{F}$-控制线：连接 $\mathcal{S}$ 与 $\mathcal{F}$（允许影响）.
- $\mathcal{S}$-控制线：连接 $\mathcal{Z}$ 与 $\mathcal{S}$（控制与泄露）.

理想协议运行结构

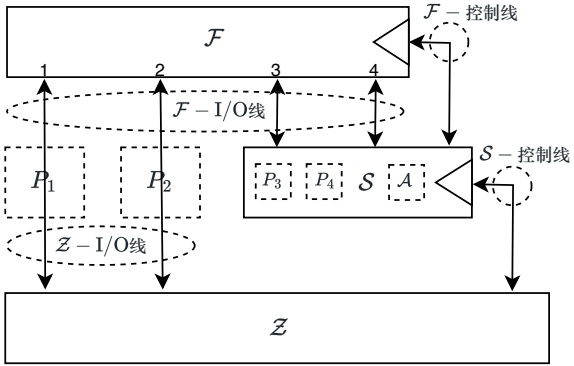


- $\mathcal{F}$ 替代网络 $\mathcal{C}$.
- 被攻陷方接入模拟器 $\mathcal{S}$.
- $\mathcal{Z}$ 与“诚实方”交互，并与“敌手” $\mathcal{S}$ 通信.

- $\mathcal{F}$-I/O 线：连接诚实方与 $\mathcal{F}$（输入/输出转发）.
- $\mathcal{F}$-控制线：连接 $\mathcal{S}$ 与 $\mathcal{F}$（允许影响）.
- $\mathcal{S}$-控制线：连接 $\mathcal{Z}$ 与 $\mathcal{S}$（控制与泄露）.

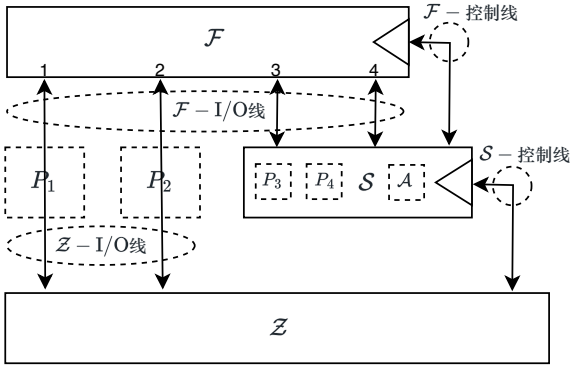
- 真实/理想世界对环境 $\mathcal{Z}$ 的接口**完全一致**.
- 核心目标：通过 $\mathcal{S}$，使理想世界对 $\mathcal{Z}$ **不可区分**于真实世界.

理想功能 $\mathcal{F}_{\text{sfe}}$



$\mathcal{F}_{\text{sfe}}$ 充当 “可信第三方”:

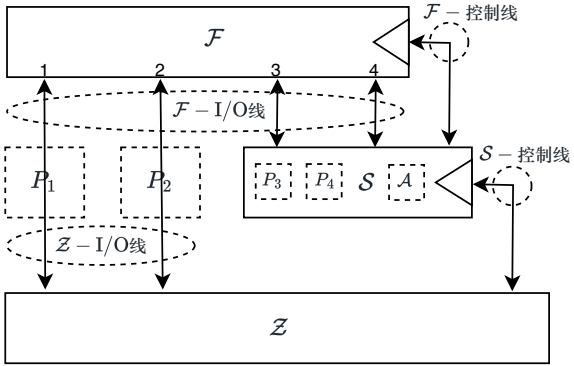
理想功能 $\mathcal{F}_{\text{sfe}}$



$\mathcal{F}_{\text{sfe}}$ 充当“可信第三方”：

- 接收输入：通过 I/O 线接收 $(\text{INPUT}, \mathit{inputID}, x)$.

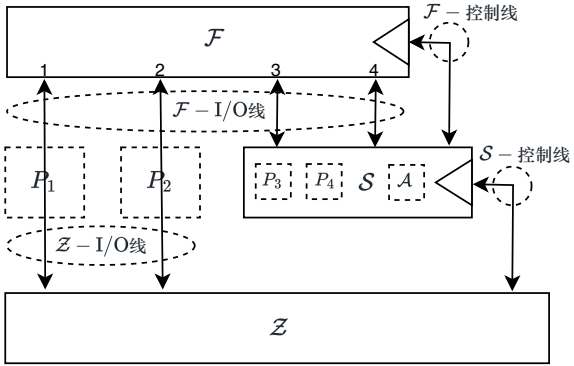
理想功能 $\mathcal{F}_{\text{sfe}}$



$\mathcal{F}_{\text{sfe}}$ 充当“可信第三方”：

- 接收输入：通过 I/O 线接收 $(\text{INPUT}, \mathit{inputID}, x)$.
- 记录输入并向模拟器发送泄露消息：$(\text{INPUT}, \mathit{inputID})$.

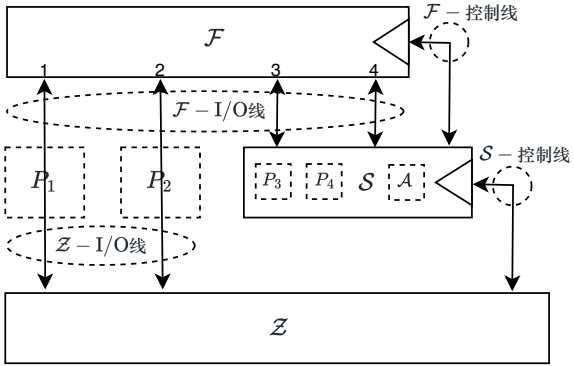
理想功能 $\mathcal{F}_{\text{sfe}}$



$\mathcal{F}_{\text{sfe}}$ 充当“可信第三方”：

- 接收输入：通过 I/O 线接收 $(\text{INPUT}, \mathit{inputID}, x)$.
- 记录输入并向模拟器发送泄露消息：$(\text{INPUT}, \mathit{inputID})$.
- 接收控制消息：模拟器发送 $(\text{OUTPUT}, \mathit{outputID}, i)$.

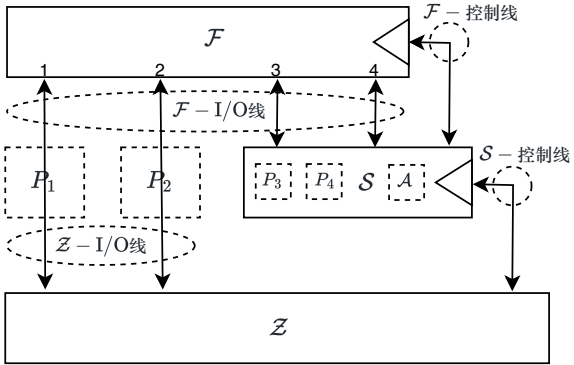
理想功能 $\mathcal{F}_{\text{sfe}}$



$\mathcal{F}_{\text{sfe}}$ 充当“可信第三方”：

- 接收输入：通过 I/O 线接收 $(\textsc{Input}, \mathit{inputID}, x)$.
- 记录输入并向模拟器发送泄露消息：$(\textsc{Input}, \mathit{inputID})$.
- 接收控制消息：模拟器发送 $(\textsc{Output}, \mathit{outputID}, i)$.
- 计算输出 y 并发送 $(\textsc{Output}, \mathit{outputID}, y)$ 给 P_i（或被攻陷方对应的 $\mathcal{S}$）.

理想功能 $\mathcal{F}_{\text{sfe}}$



$\mathcal{F}_{\text{sfe}}$ 充当“可信第三方”：

- 接收输入：通过 I/O 线接收 $(\textsc{Input}, \mathit{inputID}, x)$.
- 记录输入并向模拟器发送泄露消息：$(\textsc{Input}, \mathit{inputID})$.
- 接收控制消息：模拟器发送 $(\textsc{Output}, \mathit{outputID}, i)$.
- 计算输出 y 并发送 $(\textsc{Output}, \mathit{outputID}, y)$ 给 P_i（或被攻陷方对应的 $\mathcal{S}$）.

注 1：若函数是概率性的，$\mathcal{F}_{\text{sfe}}$ 需**自行生成随机性**并参与计算.

注 2：$\mathcal{F}_{\text{sfe}}$ 允许敌手控制诚实方的输出接收时机和条件，不保证所有诚实方都能收到输出.

运行时间考虑

良好环境

如果环境 $\mathcal{Z}$ 的运行时间是关于安全参数的**多项式**（可依赖于 $\mathcal{Z}$），则称 $\mathcal{Z}$ 是良好的.

运行时间考虑

良好环境

如果环境 $\mathcal{Z}$ 的运行时间是关于安全参数的**多项式**（可依赖于 $\mathcal{Z}$），则称 $\mathcal{Z}$ 是良好的.

敌手（模拟器）与协议（理想功能）的兼容性

在任意良好环境 $\mathcal{Z}$ 下，诚实方 P_i（理想功能 $\mathcal{F}$）与敌手 $\mathcal{A}$（模拟器 $\mathcal{S}$）的**总运行时间**以压倒性概率是关于安全参数与 $\mathcal{Z}$ 发送消息长度的多项式（可依赖于 $\Pi/\mathcal{F},\mathcal{A}/\mathcal{S}$），则称 $\mathcal{A}/\mathcal{S}$ 与协议 Π 兼容.

运行时间考虑

良好环境

如果环境 $\mathcal{Z}$ 的运行时间是关于安全参数的**多项式**（可依赖于 $\mathcal{Z}$），则称 $\mathcal{Z}$ 是良好的.

敌手（模拟器）与协议（理想功能）的兼容性

在任意良好环境 $\mathcal{Z}$ 下，诚实方 P_i（理想功能 $\mathcal{F}$）与敌手 $\mathcal{A}$（模拟器 $\mathcal{S}$）的**总运行时间**以压倒性概率是关于安全参数与 $\mathcal{Z}$ 发送消息长度的多项式（可依赖于 $\Pi/\mathcal{F},\mathcal{A}/\mathcal{S}$），则称 $\mathcal{A}/\mathcal{S}$ 与协议 Π 兼容.

高效协议

如果**平凡敌手**（只会忠实地遵循 $\mathcal{Z}$ 的指示转发消息的“路由器”）与协议 Π 兼容，则称 Π 高效.

运行时间考虑

良好环境

如果环境 $\mathcal{Z}$ 的运行时间是关于安全参数的**多项式**（可依赖于 $\mathcal{Z}$），则称 $\mathcal{Z}$ 是良好的.

敌手（模拟器）与协议（理想功能）的兼容性

在任意良好环境 $\mathcal{Z}$ 下，诚实方 P_i（理想功能 $\mathcal{F}$）与敌手 $\mathcal{A}$（模拟器 $\mathcal{S}$）的**总运行时间**以压倒性概率是关于安全参数与 $\mathcal{Z}$ 发送消息长度的多项式（可依赖于 $\Pi/\mathcal{F}, \mathcal{A}/\mathcal{S}$），则称 $\mathcal{A}/\mathcal{S}$ 与协议 Π 兼容.

高效协议

如果**平凡敌手**（只会忠实地遵循 $\mathcal{Z}$ 的指示转发消息的“路由器”）与协议 Π 兼容，则称 Π 高效.

- 高效协议下，所有机器的总运行时间仍是多项式.
- 高效协议允许协议处理来自环境的**无限数量请求**.

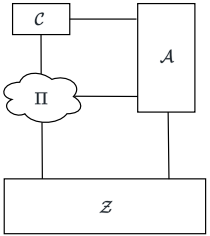
理想功能的 UC-安全实现

UC-安全实现（针对特定敌手）

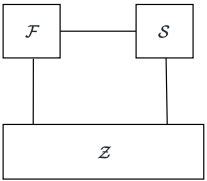
设 $\mathcal{A}$ 与协议 Π 兼容. 若存在与 $\mathcal{F}$ 兼容的模拟器 $\mathcal{S}$，使得对每个良好环境 $\mathcal{Z}$，都有

- $\mathrm{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$：$\mathcal{Z}$ 在真实世界中与协议 Π、敌手 $\mathcal{A}$ 交互后输出的随机变量
- $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$：$\mathcal{Z}$ 在理想世界中与理想功能 $\mathcal{F}$、模拟器 $\mathcal{S}$ 交互后输出的随机变量

不可区分，则称 Π 对敌手 $\mathcal{A}$ 安全实现了理想功能 $\mathcal{F}$.



真实世界

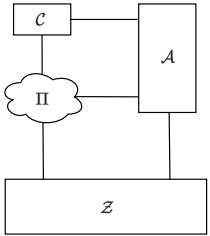


理想世界

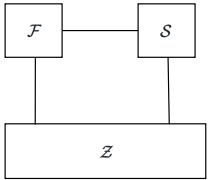
理想功能的 UC-安全实现

UC-安全实现（对所有敌手）

如果对于每个与 Π 兼容的敌手 $\mathcal{A}$，Π 都对于该敌手 $\mathcal{A}$ 安全地实现了的理想功能 $\mathcal{F}$，那么我们说协议 Π UC-安全实现了理想功能 $\mathcal{F}$.



真实世界



理想世界

平凡敌手的完备性

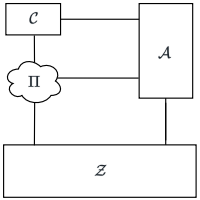
平凡敌手的完备性

设 Π 是一个高效的协议. 如果 Π 对于平凡敌手安全地实现了理想功能 $\mathcal{F}$,
则 Π UC-安全实现了 $\mathcal{F}$.

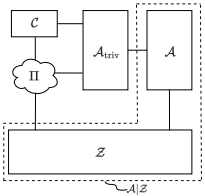
平凡敌手的完备性

平凡敌手的完备性

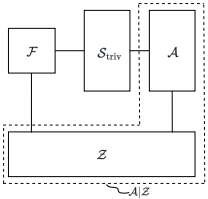
设 Π 是一个高效的协议. 如果 Π 对于平凡敌手安全地实现了理想功能 $\mathcal{F}$, 则 Π UC-安全实现了 $\mathcal{F}$.



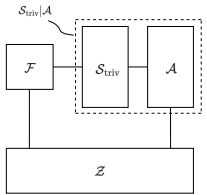
$\mathrm{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$



$\mathrm{EXEC}_{\Pi,\mathcal{A}_{\mathrm{triv}},\mathcal{A}|\mathcal{Z}}$



$\mathrm{EXEC}_{\mathcal{F},\mathcal{S}_{\mathrm{triv}},\mathcal{A}|\mathcal{Z}}$

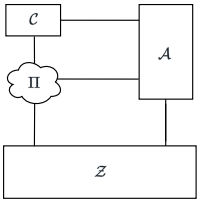


$\mathrm{EXEC}_{\mathcal{F},\mathcal{S}_{\mathrm{triv}}|\mathcal{A},\mathcal{Z}}$

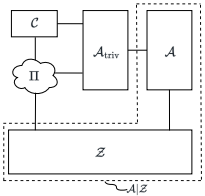
平凡敌手的完备性

平凡敌手的完备性

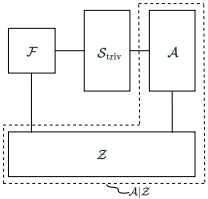
设 Π 是一个高效的协议. 如果 Π 对于平凡敌手安全地实现了理想功能 $\mathcal{F}$, 则 Π UC-安全实现了 $\mathcal{F}$.



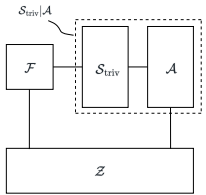
$\mathrm{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$



$\mathrm{EXEC}_{\Pi,\mathcal{A}_{\mathrm{triv}},\mathcal{A}|\mathcal{Z}}$



$\mathrm{EXEC}_{\mathcal{F},\mathcal{S}_{\mathrm{triv}},\mathcal{A}|\mathcal{Z}}$



$\mathrm{EXEC}_{\mathcal{F},\mathcal{S}_{\mathrm{triv}}|\mathcal{A},\mathcal{Z}}$

$$\mathrm{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \approx \mathrm{EXEC}_{\Pi,\mathcal{A}_{\mathrm{triv}},\mathcal{A}|\mathcal{Z}} \approx \mathrm{EXEC}_{\mathcal{F},\mathcal{S}_{\mathrm{triv}},\mathcal{A}|\mathcal{Z}} \approx \mathrm{EXEC}_{\mathcal{F},\mathcal{S}_{\mathrm{triv}}|\mathcal{A},\mathcal{Z}}$$

并发组合的安全性

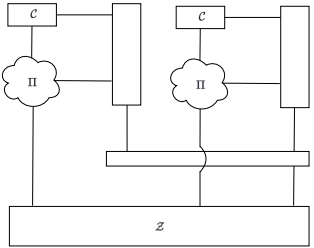
并发组合定理

如果 Π 在单实例设置中是一个高效的协议，那么它在多实例设置中仍然是高效的. 此外，如果 Π 在单实例设置中 UC-安全实现了 $\mathcal{F}$，那么它在多实例设置中也 UC-安全实现了 $\mathcal{F}$.

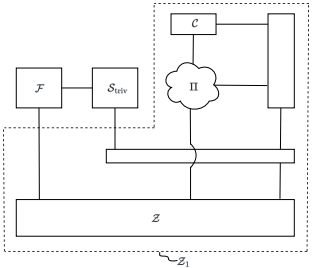
并发组合的安全性

并发组合定理

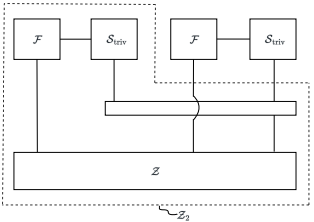
如果 Π 在单实例设置中是一个高效的协议，那么它在多实例设置中仍然是高效的. 此外，如果 Π 在单实例设置中 UC-安全实现了 $\mathcal{F}$，那么它在多实例设置中也 UC-安全实现了 $\mathcal{F}$.



真实世界



混合世界



理想世界

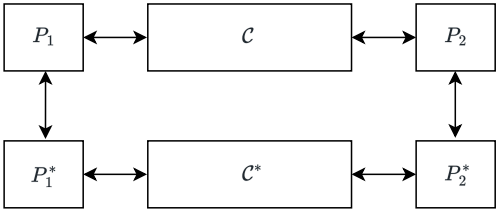
子协议组合的安全性

子协议组合定理

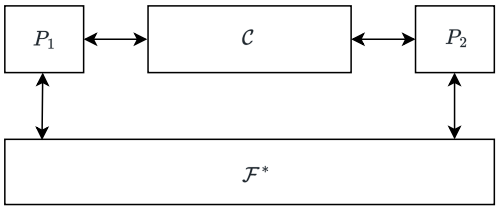
如果以下条件均成立：

- Π^* 是一个高效的协议，它安全地实现了理想功能 $\mathcal{F}^*$.
- Π 是一个高效的协议，它使用 Π^* 作为子协议.
- 混合协议 $\Pi_{\mathcal{F}^*}$(其中理想功能 $\mathcal{F}^*$ 取代了子协议 Π^*)UC-安全实现了 $\mathcal{F}$.

那么 Π UC-安全实现了 $\mathcal{F}$.

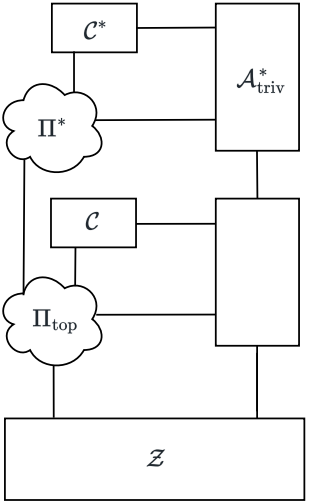


混合协议 Π

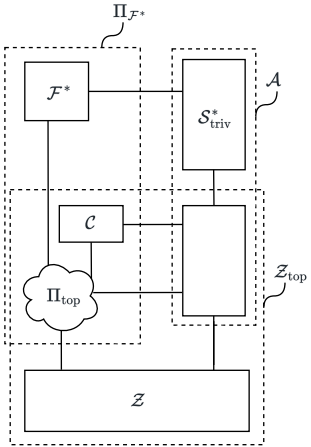


混合协议 $\Pi_{\mathcal{F}^*}$

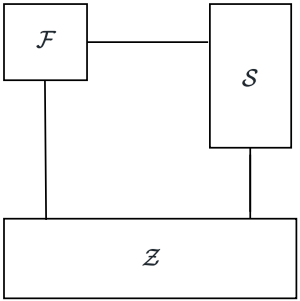
子协议组合的安全性



真实世界



混合世界



理想世界

安全实现的传递性

安全实现的传递性

假设 Π, Π', Π'' 是高效的混合协议. 如果 Π UC-安全实现了 Π' 且 Π' UC-安全实现了 Π'', 则 Π UC-安全实现了 Π''.

安全实现的传递性

安全实现的传递性

假设 Π, Π', Π'' 是高效的混合协议. 如果 Π UC-安全实现了 Π' 且 Π' UC-安全实现了 Π'', 则 Π UC-安全实现了 Π''.

- 若 Π 安全实现 Π'，则对任意与 Π 兼容的敌手 $\mathcal{A}$，存在与 Π' 兼容的敌手 $\mathcal{A}'$，使得

$$\mathrm{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \approx \mathrm{EXEC}_{\Pi',\mathcal{A}',\mathcal{Z}}.$$

安全实现的传递性

安全实现的传递性

假设 Π, Π', Π'' 是高效的混合协议. 如果 Π UC-安全实现了 Π' 且 Π' UC-安全实现了 Π'', 则 Π UC-安全实现了 Π''.

- 若 Π 安全实现 Π'，则对任意与 Π 兼容的敌手 $\mathcal{A}$，存在与 Π' 兼容的敌手 $\mathcal{A}'$，使得

$$\mathrm{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \approx \mathrm{EXEC}_{\Pi',\mathcal{A}',\mathcal{Z}}.$$

- 若 Π' 安全实现 Π''，则对任意与 Π' 兼容的敌手 $\mathcal{A}'$，存在与 Π'' 兼容的敌手 $\mathcal{A}''$，使得

$$\mathrm{EXEC}_{\Pi',\mathcal{A}',\mathcal{Z}} \approx \mathrm{EXEC}_{\Pi'',\mathcal{A}'',\mathcal{Z}}.$$

安全实现的传递性

安全实现的传递性

假设 Π, Π', Π'' 是高效的混合协议. 如果 Π UC-安全实现了 Π' 且 Π' UC-安全实现了 Π'', 则 Π UC-安全实现了 Π''.

- 若 Π 安全实现 Π'，则对任意与 Π 兼容的敌手 $\mathcal{A}$，存在与 Π' 兼容的敌手 $\mathcal{A}'$，使得

$$\mathrm{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \approx \mathrm{EXEC}_{\Pi',\mathcal{A}',\mathcal{Z}}.$$

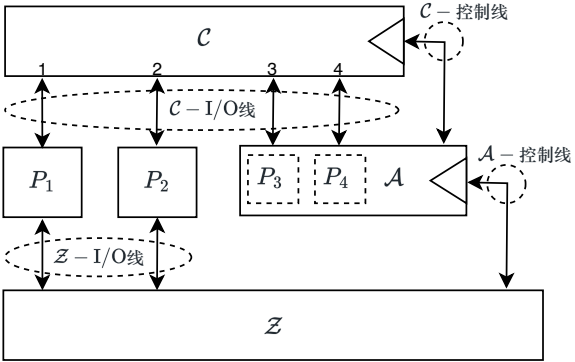
- 若 Π' 安全实现 Π''，则对任意与 Π' 兼容的敌手 $\mathcal{A}'$，存在与 Π'' 兼容的敌手 $\mathcal{A}''$，使得

$$\mathrm{EXEC}_{\Pi',\mathcal{A}',\mathcal{Z}} \approx \mathrm{EXEC}_{\Pi'',\mathcal{A}'',\mathcal{Z}}.$$

- 由传递性可得：对任意与 Π 兼容的敌手 $\mathcal{A}$，存在与 Π'' 兼容的敌手 $\mathcal{A}''$，使得

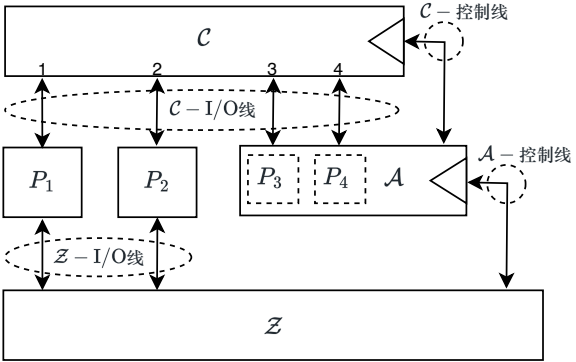
$$\mathrm{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \approx \mathrm{EXEC}_{\Pi'',\mathcal{A}'',\mathcal{Z}}. \quad \square$$

受限框架的真实世界



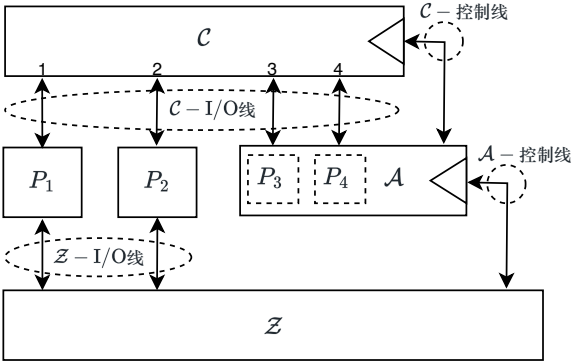
- 被攻陷方报告:
  - 从通信网络接收到的所有消息.
  - 其生成的随机比特串.

受限框架的真实世界



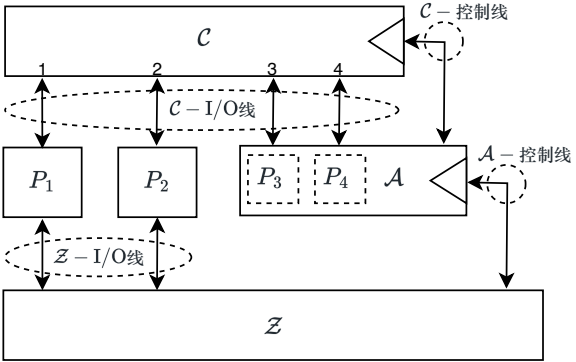
- 被攻陷方报告：
  - 从通信网络接收到的所有消息.
  - 其生成的随机比特串.
- 合法敌手的动作：
  - 立即把控制权交还给报告方；或
  - 向环境发送若干报告后交还控制权.

受限框架的真实世界



- 被攻陷方报告：
  - 从通信网络接收到的所有消息.
  - 其生成的随机比特串.
- 合法敌手的动作：
  - 立即把控制权交还给报告方；或
  - 向环境发送若干报告后交还控制权.
- 合法环境：收到敌手报告时必须立即交还控制权或停止.

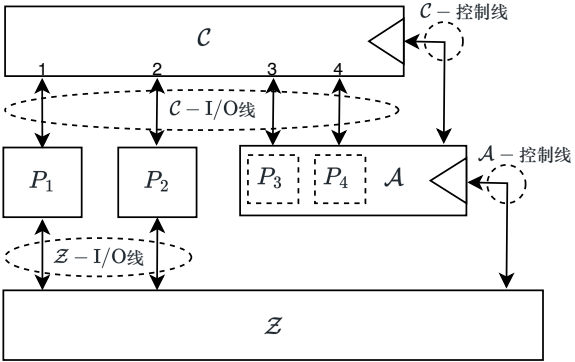
受限框架的真实世界



- 被攻陷方报告：
  - 从通信网络接收到的所有消息.
  - 其生成的随机比特串.
- 合法敌手的动作：
  - 立即把控制权交还给报告方；或
  - 向环境发送若干报告后交还控制权.
- 合法环境：收到敌手报告时必须立即交还控制权或停止.

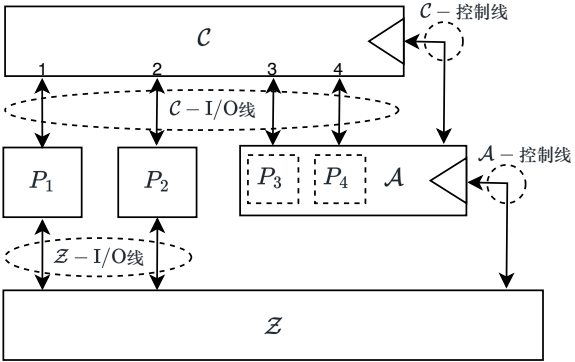
敌手与环境只能在**不中断协议流程**的前提下进行监控.

受限框架的真实世界



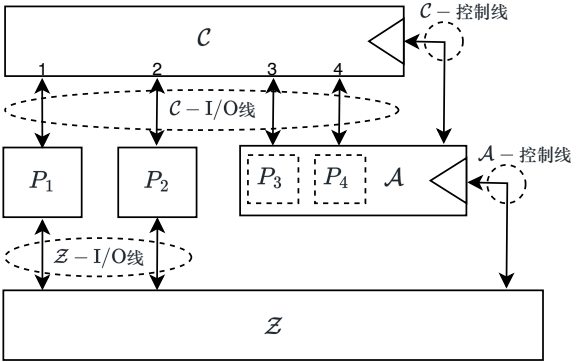
- $\mathcal{C}$ 的行为与无限制框架一致.

受限框架的真实世界



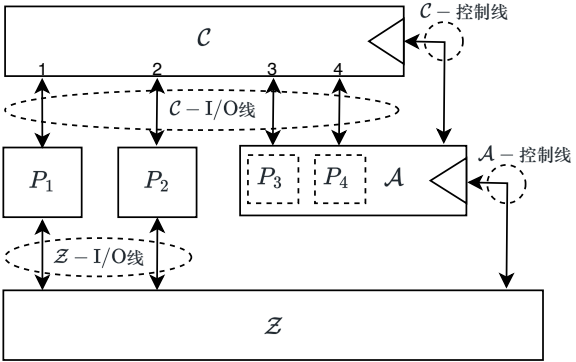
- $\mathcal{C}$ 的行为与无限制框架一致.
- 敌手仅能通过 $\mathcal{C}$-控制线交互，不能通过 $\mathcal{C}$-I/O 线直接改写消息.

受限框架的真实世界



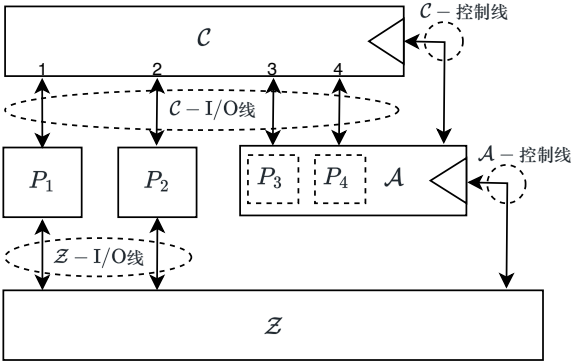
- $\mathcal{C}$ 的行为与无限制框架一致.
- 敌手仅能通过 $\mathcal{C}$-控制线交互，不能通过 $\mathcal{C}$-I/O 线直接改写消息.
- 敌手可调度消息传递，但不能得知/更改诚实方之间的消息内容.

受限框架的真实世界



- $\mathcal{C}$ 的行为与无限制框架一致.
- 敌手仅能通过 $\mathcal{C}$-控制线交互，不能通过 $\mathcal{C}$-I/O 线直接改写消息.
- 敌手可调度消息传递，但不能得知/更改诚实方之间的消息内容.
- 敌手可看到发往被攻陷方的消息与其发出的消息内容.

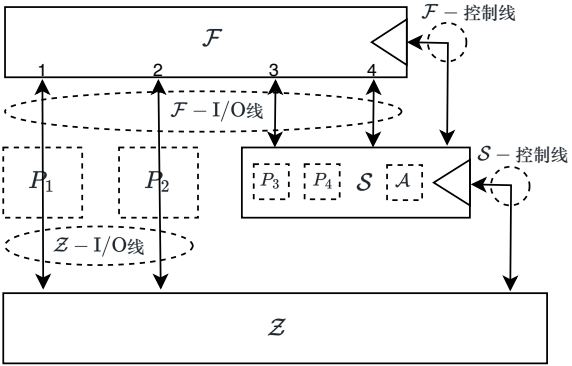
受限框架的真实世界



- $\mathcal{C}$ 的行为与无限制框架一致.
- 敌手仅能通过 $\mathcal{C}$-控制线交互，不能通过 $\mathcal{C}$-I/O 线直接改写消息.
- 敌手可调度消息传递，但不能得知/更改诚实方之间的消息内容.
- 敌手可看到发往被攻陷方的消息与其发出的消息内容.

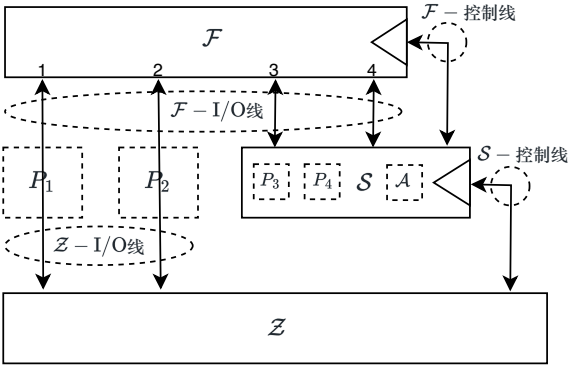
在该框架中，平凡敌手仅将报告信息原样转发给环境.

受限框架的理想世界



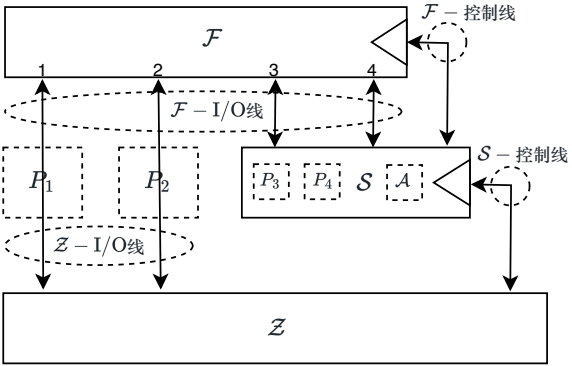
- 所有 $\mathcal{F}$-I/O 线直接连接环境与理想功能.

受限框架的理想世界



- 所有 $\mathcal{F}$-I/O 线直接连接环境与理想功能.
- 模拟器 $\mathcal{S}$ **不能更改**被攻陷方输入/输出.

受限框架的理想世界



- 所有 $\mathcal{F}$-I/O 线直接连接环境与理想功能.
- 模拟器 $\mathcal{S}$ **不能更改**被攻陷方输入/输出.
- 被攻陷方依然 “诚实但泄露”:
  - 从环境接收输入/从理想功能接收输出时，会将这些消息发送给 $\mathcal{S}$.

半诚实安全性的定义

UC-安全实现（针对特定敌手）

设 $\mathcal{A}$ 与协议 Π 兼容. 若存在与 $\mathcal{F}$ 兼容的模拟器 $\mathcal{S}$，使得对每个良好环境 $\mathcal{Z}$，都有

- $\mathrm{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$：$\mathcal{Z}$ 在真实世界中与协议 Π、敌手 $\mathcal{A}$ 交互后输出的随机变量
- $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$：$\mathcal{Z}$ 在理想世界中与理想功能 $\mathcal{F}$、模拟器 $\mathcal{S}$ 交互后输出的随机变量

不可区分，则称 Π 对敌手 $\mathcal{A}$ 安全实现了理想功能 $\mathcal{F}$.

半诚实安全性的定义

UC-安全实现（针对特定敌手）

设 $\mathcal{A}$ 与协议 Π 兼容. 若存在与 $\mathcal{F}$ 兼容的模拟器 $\mathcal{S}$，使得对每个良好环境 $\mathcal{Z}$，都有

- $\mathrm{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$：$\mathcal{Z}$ 在真实世界中与协议 Π、敌手 $\mathcal{A}$ 交互后输出的随机变量
- $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$：$\mathcal{Z}$ 在理想世界中与理想功能 $\mathcal{F}$、模拟器 $\mathcal{S}$ 交互后输出的随机变量

不可区分，则称 Π 对敌手 $\mathcal{A}$ 安全实现了理想功能 $\mathcal{F}$.

UC-安全实现（对所有敌手）

如果对于每个与 Π 兼容的敌手 $\mathcal{A}$，Π 都对于该敌手 $\mathcal{A}$ 安全地实现了的理想功能 $\mathcal{F}$，那么我们说协议 Π UC-安全实现了理想功能 $\mathcal{F}$.

半诚实安全性的定义

UC-安全实现（针对特定敌手）

设 $\mathcal{A}$ 与协议 Π 兼容. 若存在与 $\mathcal{F}$ 兼容的模拟器 $\mathcal{S}$，使得对每个良好环境 $\mathcal{Z}$，都有

- $\mathrm{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$：$\mathcal{Z}$ 在真实世界中与协议 Π、敌手 $\mathcal{A}$ 交互后输出的随机变量
- $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$：$\mathcal{Z}$ 在理想世界中与理想功能 $\mathcal{F}$、模拟器 $\mathcal{S}$ 交互后输出的随机变量

不可区分，则称 Π 对敌手 $\mathcal{A}$ 安全实现了理想功能 $\mathcal{F}$.

UC-安全实现（对所有敌手）

如果对于每个与 Π 兼容的敌手 $\mathcal{A}$，Π 都对于该敌手 $\mathcal{A}$ 安全地实现了的理想功能 $\mathcal{F}$，那么我们说协议 Π UC-安全实现了理想功能 $\mathcal{F}$.

以上定义均可直接应用到受限框架下.

半诚实安全性的效果

平凡敌手的完备性

设 Π 是一个高效的协议. 如果 Π 对于平凡敌手安全地实现了理想功能 $\mathcal{F}$, 则 Π UC-安全实现了 $\mathcal{F}$.

半诚实安全性的效果

平凡敌手的完备性

设 Π 是一个高效的协议. 如果 Π 对于平凡敌手安全地实现了理想功能 $\mathcal{F}$,
则 Π UC-安全实现了 $\mathcal{F}$.

并发组合定理

如果 Π 在单实例设置中是一个高效的协议，那么它在多实例设置中仍然是高效的. 此外，
如果 Π 在单实例设置中 UC-安全实现了 $\mathcal{F}$，那么它在多实例设置中也 UC-安全实现了 $\mathcal{F}$.

半诚实安全性的效果

平凡敌手的完备性

设 Π 是一个高效的协议. 如果 Π 对于平凡敌手安全地实现了理想功能 $\mathcal{F}$,
则 Π UC-安全实现了 $\mathcal{F}$.

并发组合定理

如果 Π 在单实例设置中是一个高效的协议，那么它在多实例设置中仍然是高效的. 此外，
如果 Π 在单实例设置中 UC-安全实现了 $\mathcal{F}$，那么它在多实例设置中也 UC-安全实现了 $\mathcal{F}$.

以上结论能在受限框架中以完全相同的方式证明.

半诚实安全性的效果

子协议组合定理

如果以下条件均成立：

- Π^* 是一个高效的协议，它安全地实现了理想功能 $\mathcal{F}^*$.
- Π 是一个高效的协议，它使用 Π^* 作为子协议.
- 混合协议 $\Pi_{\mathcal{F}^*}$(其中理想功能 $\mathcal{F}^*$ 取代了子协议 Π^*)UC-安全实现了 $\mathcal{F}$.

那么 Π UC-安全实现了 $\mathcal{F}$.

半诚实安全性的效果

子协议组合定理

如果以下条件均成立：

- Π^* 是一个高效的协议，它安全地实现了理想功能 $\mathcal{F}^*$.
- Π 是一个高效的协议，它使用 Π^* 作为子协议.
- 混合协议 $\Pi_{\mathcal{F}^*}$(其中理想功能 $\mathcal{F}^*$ 取代了子协议 Π^*)UC-安全实现了 $\mathcal{F}$.

那么 Π UC-安全实现了 $\mathcal{F}$.

安全实现的传递性

假设 Π, Π', Π'' 是高效的混合协议. 如果 Π UC-安全实现了 Π' 且 Π' UC-安全实现了 Π''，则 Π UC-安全实现了 Π''.

半诚实安全性的效果

子协议组合定理

如果以下条件均成立：

- Π^* 是一个高效的协议，它安全地实现了理想功能 $\mathcal{F}^*$.
- Π 是一个高效的协议，它使用 Π^* 作为子协议.
- 混合协议 $\Pi_{\mathcal{F}^*}$(其中理想功能 $\mathcal{F}^*$ 取代了子协议 Π^*)UC-安全实现了 $\mathcal{F}$.

那么 Π UC-安全实现了 $\mathcal{F}$.

安全实现的传递性

假设 Π, Π', Π'' 是高效的混合协议. 如果 Π UC-安全实现了 Π' 且 Π' UC-安全实现了 Π''，则 Π UC-安全实现了 Π''.

以上结论能在受限框架中以完全相同的方式证明.

不同类型的攻陷方式

- **静态攻陷 (Static Corruption)（前文采用的方式）**：
  - 协议开始前，$\mathcal{Z}$ 必须决定攻陷哪些参与方.

不同类型的攻陷方式

- **静态攻陷 (Static Corruption)（前文采用的方式）**：
  - 协议开始前，$\mathcal{Z}$ 必须决定攻陷哪些参与方.
- **适应性攻陷 (Adaptive Corruption)**:
  - $\mathcal{Z}$ 可以在协议执行过程中，动态决定攻陷某个参与方.
  - 模拟器更难设计（需要模拟被攻陷方的内部历史状态）.

不同类型的攻陷方式

- **静态攻陷 (Static Corruption)（前文采用的方式）**：
  - 协议开始前，$\mathcal{Z}$ 必须决定攻陷哪些参与方.
- **适应性攻陷 (Adaptive Corruption)**:
  - $\mathcal{Z}$ 可以在协议执行过程中，动态决定攻陷某个参与方.
  - 模拟器更难设计（需要模拟被攻陷方的内部历史状态）.
- **主动安全/可移动攻陷 (Proactive/Mobile)**:
  - 参与方被攻陷后可以恢复（需要安全擦除、周期性刷新）.

思考题

1. 通用可组合框架中的“环境” $\mathcal{Z}$ 在现实中包括什么？
2. 在通用可组合框架中，模拟器 $\mathcal{S}$ 需要完成哪两个关键任务？这些任务是如何反映输入的隐私性和输出结果的正确性的？
3. 在通用可组合框架中，一个协议的子协议实例或者一个协议所调用的理想功能能否被其他协议调用？为什么？

Q & A