Compressing Vector OLE

贡献

- 论文的目的是构造一个两方协议，为双方各自生成随机数，且双方持有的随机数满足VOLE关系。
- 这篇论文提出了一个random VOLE correlation generator with good concrete efficiency.主要思路是协议双方经tiny interaction获得small seed, then via a deterministic local expansion to generate VOLE correlation with no interaction

关键技术

- VOLE generator 基于多点函数秘密共享和解码困难的线性编码，安全性依赖于LPN假设，是抗量子安全的。

预备知识

VOLE基础知识

intuition: VOLE是一个两方协议，允许在不泄露双方输入的情况下，一方了解到另一方输入向量的特定线性组合。

- OLE functionary:
  - syntax: the OLE functionary takes a pair of values(u,v) $\in \mathbb{F}$ from P_0, and a value x $\in \mathbb{F}$ from P_1, It outputs u · x+v to P_1;
  - security: the P_0 can not get any information about x and u · x+v, the P_1 can not get any information about u, v.
- VOLE functionary:
  - syntax: the VOLE functionality takes a pair of vectors (**u**, **v**) $\in \mathbb{F}^{m} \times \mathbb{F}^{m}$ from P_0 and a scalar $x \in \mathbb{F}$ from P_1. It outputs **w** = **u** · x + **v** to P_1.
  - security: the P_0 gets no information about **w** and x, the P_1 gets no information about **u** and **v**;
- Random VOLE functionary:
  - syntax: for random VOLE, the functionality chooses randomly a pair of vector of (R_u, R_v) $\in \mathbb{F}^n \times \mathbb{F}^n$, a scalar $R_x \in \mathbb{F}$ and then outputs (R_u, R_v) to P_0 and outputs ($R_x, R_u R_x + R_v$) to P_1.
- 存在标准方法由Random VOLE 构造 Standard VOLE.

FSS

- intuition: a function secret sharing (FSS) scheme splits a function $f : I \rightarrow G$ into two functions f_0 and f_1 such that $f_0(x) + f_1(x) = f(x)$ for every input x, and each f_b computationally hides f .
- formal definition:

Definition 2.1 (Adapted from [19]). A 2-party *function secret sharing* (FSS) scheme for a class of functions $\mathcal{F} = \{f : I \to \mathbb{G}\}$ with input domain I and output domain an abelian group $(\mathbb{G}, +)$, is a pair of PPT algorithms FSS = (FSS.Gen, FSS.Eval) with the following syntax:

- $\mathsf{FSS.Gen}(1^\lambda, f)$, given security parameter λ and description of a function $f \in \mathcal{F}$, outputs a pair of keys (K_0, K_1);
- $\mathsf{FSS.Eval}(b, K_b, x)$, given party index $b \in \{0,1\}$, key K_b, and input $x \in I$, outputs a group element $y_b \in \mathbb{G}$.

Given an allowable leakage function Leak : $\{0,1\}^* \to \{0,1\}^*$, the scheme FSS should satisfy the following requirements:

- **Correctness.** For any $f : I \to \mathbb{G}$ in $\mathcal{F}$ and $x \in I$, we have $\Pr[(K_0, K_1) \stackrel{R}{\leftarrow} \mathsf{FSS.Gen}(1^\lambda, f) : \sum_{b \in \{0,1\}} \mathsf{FSS.Eval}(b, K_b, x) = f(x)] = 1$.
- **Security.** For any $b \in \{0,1\}$, there exists a PPT simulator Sim such that for any polynomial-size function sequence $f_\lambda \in \mathcal{F}$, the distributions $\{(K_0, K_1) \stackrel{R}{\leftarrow} \mathsf{FSS.Gen}(1^\lambda, f_\lambda) : K_b\}$ and $\{K_b \stackrel{R}{\leftarrow} \mathsf{Sim}(1^\lambda, \mathsf{Leak}(f_\lambda))\}$ are computationally indistinguishable.

  - FSS.FullEval (b,K_b) , 给定$b \in \{0,1\}$, key K_b ,输出 f_b 在 I 上的所有值的向量;
  - 正确性保证对于每一个输入，函数的秘密分享能共同计算出原始的函数值;
  - 安全性保证了FSS生成的子密钥并不能透露关于函数 f 的其他信息（除了允许泄露的信息之外）

DPF(Distributed point functions)

- 单点函数 $f_{\alpha,\beta}$: $\{0,1\}^\ell \to \mathbb{G}$ which satisfy $f_{\alpha,\beta}(\alpha) = \beta$, and $f_{\alpha,\beta}(x) = 0$ for any $x \neq \alpha$.
- 多点函数: 是单点函数自然的拓展，表示在定义域的某个子集内有特定的取值，其他x函数取值为0.

  Definition 2.3 (Multi-Point Function). An (n, t)-multi-point function over an abelian group $(\mathbb{G}, +)$ is a function $f_{S,\vec{y}} : [n] \to \mathbb{G}$, where $S = \{s_1, \cdots, s_t\}$ is a subset of $[n]$ of size t, $\vec{y} = (y_1, \cdots, y_t) \in \mathbb{G}^{kt}$, and $f_{S,\vec{y}}(s_i) = y_i$ for any $i \in [t]$, and $f_{S,y}(x) = 0$ for any $x \in [n] \setminus S$.

  - 关于用$[n] = 2^\ell$ instead of $\{0,1\}^\ell$来表示定义域，是整数表示相比于二进制更方便用作索引。
- DPF是对于单点函数的FSS，后续方案设计需要针对多点函数的秘密共享（MPFSS）

MPFSS

- 多点函数（t-point）是由在t个不同点取值的单点函数之和：
  $f_{S,\vec{y}} = \sum_{i=1}^{t} f_{s_i,y_i}$,其中 f_{s_i,y_i}是定义在[n]上的单点函数；
- 对于单点函数进行FSS得到DPF，对t个单点函数进行FSS得到多点函数的秘密分享（MPFSS）
- An (n,t) - MPFSS for a multi-point function $f_{S,\vec{y}} : [n] \to \mathbb{G}$ can be
  readily constructed using *t* invocations to a DPF.

- MPFSS.Gen($1^\lambda, f_{S,\vec{y}}$): denoting $s_1, \cdots, s_t$ (an arbitrary ordering of) the elements of S, for any $i \leq t$, compute $(K_0^i, K_1^i) \xleftarrow{R}$ DPF.Gen($1^\lambda, f_{s_i,y_i}$), where f_{s_i,y_i} is the point function over $\mathbb{G}$ which evaluates to y_i on s_i and to 0 otherwise. Output $(K_0, K_1) \leftarrow ((K_0^i)_{i\leq t}, (K_1^i)_{i\leq t})$.
- MPFSS.Eval(σ, K_σ, x): parse K_σ as $(K_\sigma^i)_{i\leq t}$ and compute $z_\sigma \leftarrow \sum_{i=1}^{t}$ DPF.Eval(σ, K_σ^i, x).
  - MPFSS.Eval赋予的功能是允许计算函数在x处的第 σ 个 秘密分享值。$\sigma \in \{0,1\}$
  - MPFSS.FullEval(σ, K_σ)赋予的功能是计算函数在整个定义域上的取值的第 σ 个秘密分享，计算结果为一个n维向量，其中t个坐标处的值为t个单点函数的第 σ 个秘密分享，其他值为0；

    我们有MPFSS.FullEval($0, K_0$) + MPFSS.FullEval($1, K_1$) = $[0, 0.. . y_1, 0, \ldots y_2, \ldots, y_t, 0, \ldots]$ (结果为n维向量，其中t个取值$y_1 - y_t$)根据$[s_1, s_2, \ldots, s_t]$索引值分布在n维向量中，在文中索引值是随即生成，利用随机性保证方案的安全性

LPN假设

Definition 2.4. Let C be a probabilistic code generation algorithm such that C($k, q, \mathbb{F}$) outputs (a description of) a matrix $A \in \mathbb{F}^{k\times q}$. For dimension $k = k(\lambda)$, number of queries (or block length) $q = q(\lambda)$, and noise rate $r = r(\lambda)$, the LPN(k, q, r) assumption with respect to C states that for any polynomial-time non-uniform adversary $\mathcal{A}$, it holds that

$$\Pr[\mathbb{F} \leftarrow \mathcal{A}(1^\lambda), A \xleftarrow{R} \mathrm{C}(k, q, \mathbb{F}), \vec{e} \xleftarrow{R} \mathrm{Ber}_r(\mathbb{F})^q,$$
$$\vec{s} \xleftarrow{R} \mathbb{F}^k, \vec{b} \leftarrow \vec{s} \cdot A + \vec{e} : \mathcal{A}(A, \vec{b}) = 1]$$
$$\approx \Pr[\mathbb{F} \leftarrow \mathcal{A}(1^\lambda), A \xleftarrow{R} \mathrm{C}(k, q, \mathbb{F}), \vec{b} \xleftarrow{R} \mathbb{F}^q : \mathcal{A}(A, \vec{b}) = 1].$$

- $Ber_r(\mathbb{F})$:论文的描述是 sampling a uniformly random element of F with probability r, and 0 with probability 1 − r，因此LPN的s和噪声向量，矩阵都是定义在任意一个有限域$\mathbb{F}$ 上，此处的e的分布是类伯努利分布，e中每个坐标为非零值的概率为r，而非零值则是指从$\mathbb{F}$ 中随机取样。
- LPN假设是说：给定随机矩阵A和q维向量b，PPT, non-uniform adversary 无法区分随机向量和 $\vec{s} \cdot A + \vec{e}$；
- LPN假设对于e 的要求，the noise rate of e is constant, such that noise rate is not adequately small to guarantee security.
- dual version of LPN assumption:

  难以区分$\vec{e} \cdot B$ 和一个随机向量，$\vec{e}$ 是噪音向量，B 是矩阵 A 的校验举证，such that $A \cdot B = 0$. The equivalence to LPN follows immediately from the relation $\vec{e} \cdot B = (\vec{s} \cdot A + \vec{e}) \cdot B$

  - dual version of LPN assumption相较于LPN assumption的优势

    基于dual LPN实现的generator可以实现任意多项式的expansion factor，而基于general LPN假设实现的generator只能实现次二次的expansion factor（差异在于对LPN假设的攻击限制了noise rate的大小，而noise rate则限制了expansion factor)

主协议构造

方案构造

- intuition

  论文利用small vole generator,然后利用矩阵将其编码成为long vole generator，利用LPN假设，将long vole generator加上噪音，获得long random vole generator，噪音由 sparse vole generator实现。
- 安全性定义
  - 定义一般VOLE：

    对于P_0，给定(u,v)，不知道P_1拥有哪对(x,w)，只知道P_1拥有的(x,w)满足w=ux+v;

    对于P_1，给定(x,w)，不知道P_0拥有哪对(u,v)，只知道P_0拥有的(u,v)满足w=ux+v;
  - 定义Random VOLE:

    对于P_0，给定(u,v)， P_1拥有的(x,w)与(R_x, R_w)无法区分；

    对于P_1，给定(x,w)，P_0拥有的(u,v)与(R_u, R_v)无法区分；
  - 定义 pseudorandom VOLE relation:

    对于P_0，给定$seed_0$, P_0可以本地运行Expand($seed_0$)得到(u,v)，P_0无法在多项式时间内区分Expand($seed_1$)与(R_x, R_x ·u+v);

    对于P_1，给定$seed_1$，P_1可以本地运行Expand($seed_1$)得到(w)，P_1无法在多项式时间内区分Expand($seed_0$)与(R_u, w - R_u·x);

    具体来说，两方均只知道他们手中的值满足线性关系，但是无法知道对方的值$seed_\sigma$和Expand($seed_\sigma$）到这里，理解下面正式的安全性定义会简单些：

Definition 5 (Pseudorandom VOLE generator). *A pseudorandom* VOLE *generator is a pair of algorithms* (Setup, Expand) *with the following syntax:*

- $\mathsf{Setup}(1^\lambda, \mathbb{F}, n, x)$ *is a PPT algorithm that given a security parameter* λ*, field* $\mathbb{F}$*, output length* n*, and scalar* $x \in \mathbb{F}$ *outputs a pair of seeds* $(\mathsf{seed}_0, \mathsf{seed}_1)$*, where* seed_1 *includes* x*;*
- $\mathsf{Expand}(\sigma, \mathsf{seed}_\sigma)$ *is a polynomial-time algorithm that given party index* $\sigma \in \{0,1\}$ *and a seed* seed_σ*, outputs a pair* $(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{F}^n \times \mathbb{F}^n$ *if* $\sigma = 0$*, or a vector* $\boldsymbol{w} \in \mathbb{F}^n$ *if* $\sigma = 1$*;*

The algorithms $(\mathsf{Setup}, \mathsf{Expand})$ *should satisfy the following:*

- ***Correctness.*** *For any field* $\mathbb{F}$ *and* $x \in \mathbb{F}$*, for any pair* $(\mathsf{seed}_0, \mathsf{seed}_1)$ *in the image of* $\mathsf{Setup}(1^\lambda, \mathbb{F}, n, x)$ *(for some* n*), denoting* $(\boldsymbol{u}, \boldsymbol{v}) \leftarrow \mathsf{Expand}(0, \mathsf{seed}_0)$*, and* $\boldsymbol{w} \leftarrow \mathsf{Expand}(1, \mathsf{seed}_1)$*, it holds that* $\boldsymbol{u}x + \boldsymbol{v} = \boldsymbol{w}$*.*
- ***Security.*** *For any (stateful, nonuniform) polynomial-time adversary* $\mathcal{A}$*, it holds that*

$$\Pr\left[\begin{matrix}(\mathbb{F}, 1^n, x, x') \leftarrow \mathcal{A}(1^\lambda), \\ (\mathsf{seed}_0, \mathsf{seed}_1) \xleftarrow{R} \mathsf{Setup}(1^\lambda, \mathbb{F}, n, x)\end{matrix} : \mathcal{A}(\mathsf{seed}_0) = 1\right]$$
$$\approx \Pr\left[\begin{matrix}(\mathbb{F}, 1^n, x, x') \leftarrow \mathcal{A}(1^\lambda), \\ (\mathsf{seed}_0, \mathsf{seed}_1) \xleftarrow{R} \mathsf{Setup}(1^\lambda, \mathbb{F}, n, x')\end{matrix} : \mathcal{A}(\mathsf{seed}_0) = 1\right].$$

Similarly, for any (stateful, nonuniform) adversary $\mathcal{A}$*, it holds that*

$$\Pr\left[\begin{matrix}(\mathbb{F}, 1^n, x) \leftarrow \mathcal{A}(1^\lambda), \\ (\mathsf{seed}_0, \mathsf{seed}_1) \xleftarrow{R} \mathsf{Setup}(1^\lambda, \mathbb{F}, n, x), \\ (\boldsymbol{u}, \boldsymbol{v}) \leftarrow \mathsf{Expand}(0, \mathsf{seed}_0)\end{matrix} : \mathcal{A}(\boldsymbol{u}, \boldsymbol{v}, \mathsf{seed}_1) = 1\right]$$
$$\approx \Pr\left[\begin{matrix}(\mathbb{F}, 1^n, x) \leftarrow \mathcal{A}(1^\lambda), \boldsymbol{u} \xleftarrow{R} \mathbb{F}^n, \\ (\mathsf{seed}_0, \mathsf{seed}_1) \xleftarrow{R} \mathsf{Setup}(1^\lambda, \mathbb{F}, n, x), \\ \boldsymbol{w} \leftarrow \mathsf{Expand}(1, \mathsf{seed}_1), \boldsymbol{v} \leftarrow \boldsymbol{w} - \boldsymbol{u}x\end{matrix} : \mathcal{A}(\boldsymbol{u}, \boldsymbol{v}, \mathsf{seed}_1) = 1\right].$$

Informally,$\mathcal{A}$ 给定 P_0 的view，$\mathcal{A}$无法区分$seed_1$是由x还是x1生成；

$\mathcal{A}$ 给定P_1的view，$\mathcal{A}$ 无法区分运行协议后P_0得到的和随机u,v

- 方案构造：
  - define a public matrix $C_{k,n} \in \mathbb{F}^{k \times n}$
  - setup(): fixed a scalar x, randomly generate $(a, b, c) \in \mathbb{F}^k \times \mathbb{F}^k \times \mathbb{F}^k$ 其中$c = ax + b$
  - define a sparse vole generator:

    The functionary send (v_0, v_1) to P_0， $(x, v_0x + v_1)$ to P_1, v_0, v_1 are sparse means there is few non-zero in vector with dimension n.
  - setup(): randomly generate $(a, b, c) \in \mathbb{F}^k \times \mathbb{F}^k \times \mathbb{F}^k$ 其中$c = ax + b$
  - send a, b to P_0. send c, x to P_1
  - P_0 compute $a \cdot C_{k,n}, b \cdot C_{k,n}$, P_1 compute $c \cdot C_{k,n}$ （现在我们从小的k维vole correlation生成了长的n维 vole correlation，但是pseudorandom vole correlation 要求 $a \cdot C_{k,n}, b \cdot C_{k,n}, c \cdot C_{k,n}$ 是伪随机的，根据LPN假设: $a \cdot C_{k,n}$+ noise vector是pseudorandom)
  - Setup阶段分发sparse vole correlation 给协议双方
  - P_0 compute $(a \cdot C_{k,n} + v_0, b \cdot C_{k,n} + v_1)$, P_1 compute $c \cdot C_{k,n} + v_0x + v_1$
  - 现在双方从small seed(a, b, v_0, v_1)生成了long pseudorandom vole correlation($a \cdot C_{k,n} + v_0, b \cdot C_{k,n} + v_1$，$c \cdot C_{k,n} + v_0x + v_1$)

- 只剩下一个问题没有解决，那就是sparse vole generator如何得到，论文利用MPFSS来生成sparse vole generator
  - 定义a multi-point function secret sharing MPFSS = (MPFSS.Gen; MPFSS.Eval; MPFSS.FullEval). choose a random vector $\mathbf{y} \leftarrow \mathbb{F}^{\mathbf{t}}$, MPFSS是针对多点函数$f_{S,xy}$, 利用$f_{S,xy}$将多点函数分成两个函数共享，并分别发送给双方，由于多点函数只有在 S 个点处有值，其他取值均为零，可以用于构建sparse vole correlation。
- 整体构造如下所示：

> VOLE Generator G_{primal}
>
> - **Parameters:** dimension $k = k(\lambda)$, noise parameter $t = t(\lambda)$
> - **Building blocks:** a code generator $\mathbf{C}$, such that $\mathbf{C}(k, n, \mathbb{F})$ defines a public matrix $C_{k,n} \in \mathbb{F}^{k\times n}$, and a multi-point function secret sharing $\mathsf{MPFSS} = (\mathsf{MPFSS.Gen}, \mathsf{MPFSS.Eval}, \mathsf{MPFSS.FullEval})$.
> - $G_{\mathsf{primal}}.\mathsf{Setup}(1^\lambda, \mathbb{F}, n, x)$: pick a random size-t subset S of $[n]$, two random vectors $(\boldsymbol{a}, \boldsymbol{b}) \stackrel{R}{\leftarrow} \mathbb{F}^k \times \mathbb{F}^k$, and a random vector $\boldsymbol{y} \stackrel{R}{\leftarrow} \mathbb{F}^t$. Let $s_1 < s_2 < \cdots < s_t$ denote the elements of S. Set $\boldsymbol{c} \leftarrow \boldsymbol{a}x + \boldsymbol{b}$. Compute $(K_0, K_1) \stackrel{R}{\leftarrow} \mathsf{MPFSS.Gen}(1^\lambda, f_{S,x\boldsymbol{y}})$. Set $\mathsf{seed}_0 \leftarrow (\mathbb{F}, n, K_0, S, \boldsymbol{y}, \boldsymbol{a}, \boldsymbol{b})$ and $\mathsf{seed}_1 \leftarrow (\mathbb{F}, n, K_1, x, \boldsymbol{c})$. Output $(\mathsf{seed}_0, \mathsf{seed}_1)$.
> - $G_{\mathsf{primal}}.\mathsf{Expand}(\sigma, \mathsf{seed}_\sigma)$:
>   If $\sigma = 0$, parse seed_0 as $(\mathbb{F}, n, K_0, S, \boldsymbol{y}, \boldsymbol{a}, \boldsymbol{b})$. Set $\boldsymbol{\mu} \leftarrow \mathsf{spread}_n(S, \boldsymbol{y})$. Compute $\boldsymbol{\nu_0} \leftarrow \mathsf{MPFSS.FullEval}(0, K_0)$. Output $(\boldsymbol{u}, \boldsymbol{v}) \leftarrow (\boldsymbol{a} \cdot C_{k,n} + \boldsymbol{\mu}, \boldsymbol{b} \cdot C_{k,n} - \boldsymbol{\nu_0})$.
>   If $\sigma = 1$, parse seed_1 as $(\mathbb{F}, n, K_1, x, \boldsymbol{c})$. Compute $\boldsymbol{\nu_1} \leftarrow \mathsf{MPFSS.FullEval}(1, K_1)$, and set $\boldsymbol{w} \leftarrow \boldsymbol{c} \cdot C_{k,n} + \boldsymbol{\nu_1}$. Output $\boldsymbol{w}$.

安全性证明

- **Correctness.** By the MPFSS correctness, it holds that

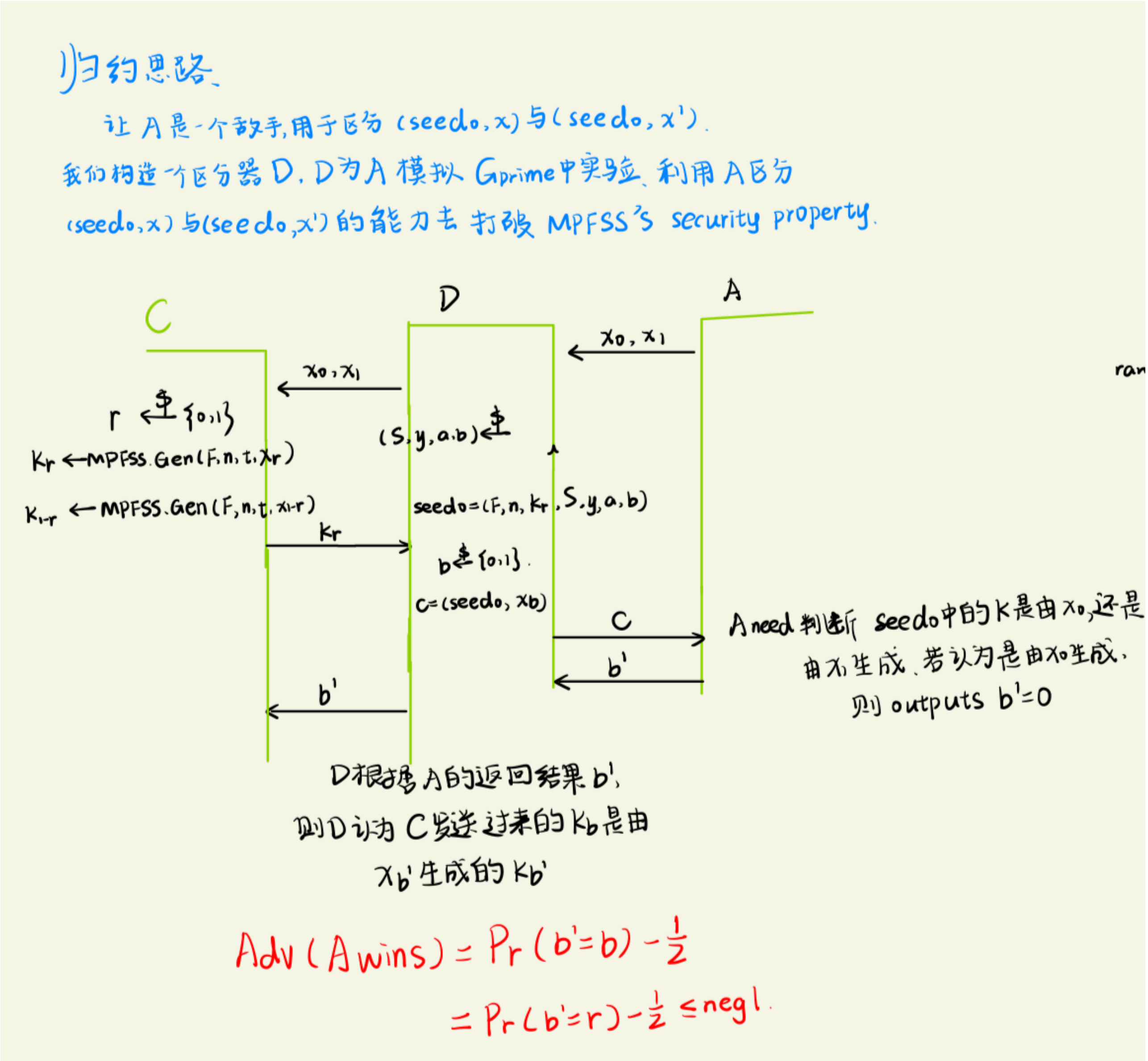
$$\begin{aligned} &\mathsf{MPFSS.FullEval}(0, K_0) \\ &\qquad + \mathsf{MPFSS.FullEval}(1, K_1) = \mathsf{spread}_n(S, x\boldsymbol{y}) = \boldsymbol{\mu}x. \end{aligned}$$

  Therefore,

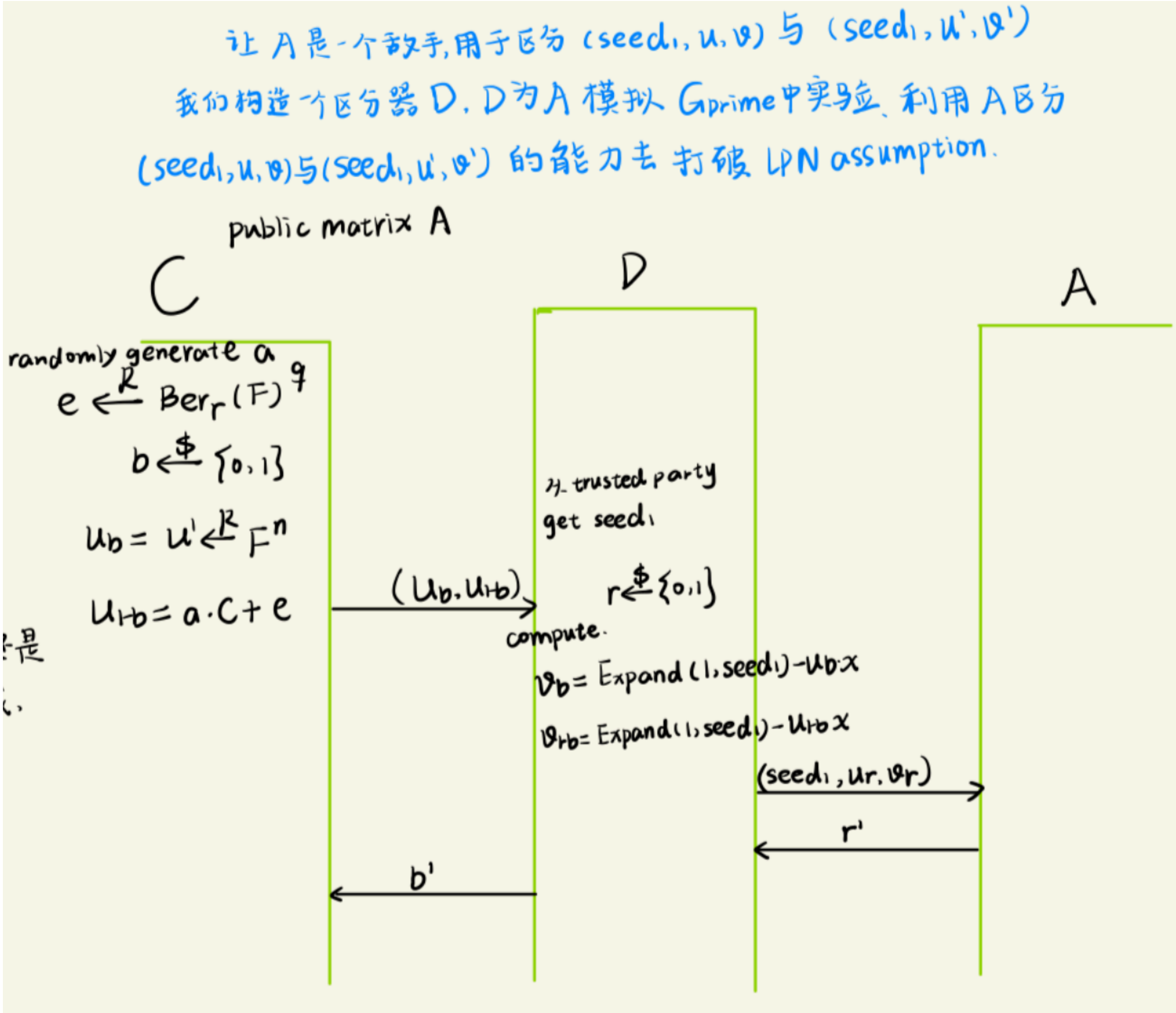
$$\begin{aligned} \boldsymbol{u}x + \boldsymbol{v} &= (\boldsymbol{a} \cdot C_{k,n} + \boldsymbol{\mu})x + \boldsymbol{b} \cdot C_{k,n} - \boldsymbol{\nu_0} \\ &= (\boldsymbol{a}x + \boldsymbol{b}) \cdot C_{k,n} + \boldsymbol{\mu}x - \mathsf{MPFSS.FullEval}(0, K_0) \\ &= \boldsymbol{c} \cdot C_{k,n} + \boldsymbol{\mu}x + \mathsf{MPFSS.FullEval}(1, K_1) - \boldsymbol{\mu}x \\ &= \boldsymbol{c} \cdot C_{k,n} + \boldsymbol{\nu}_1 = \boldsymbol{w}, \end{aligned}$$

  which concludes the proof of correctness.
- 证明security 1: $\mathcal{A}$无法区分$(seed_0, x)$和$(seed_0, x^1)$，$\mathcal{A}$只能从$seed_0$中的K_0得到关于x的信息，由于MPFSS的安全性(K_0不会暴露关于$f_{S,xy}$的信息)，保证了$\mathcal{A}$无法从K_0中的得到关于x的任何信息。

  证明思路：证明若敌手能区分$(seed_0, x)$和$(seed_0, x^1)$，则敌手能打破MPFSS安全性，归约过程如下图：



- 证明security 2: $\mathcal{A}$无法区分$(seed_1, u, v)$和$(seed_1, u^1, v^1)$
  - 是运行 pseudorandom vole correlation生成，v可以由u生成，$v = b \cdot C_{k,n} + v_0 = b \cdot C_{k,n} + v_1 - \mu x = c \cdot C_{k,n} + v_1 - (a \cdot C_{k,n} + \mu)x = w - ux$
  - 因此security 2转换为$\mathcal{A}$无法区分$(seed_1, u, Expand(1, seed_1) - ux)$和$(seed_1, u^1, Expand(1, seed_1) - u^1 x)$，这两种分布的不同只与$u$有关，因此证明$\mathcal{A}$无法区分$u = a \cdot C_{k,n} + \mu$和$u^1 \xleftarrow{R} \mathbb{F}^n$，即可证明security 2;
- 具体思路:
  - $seed_1$ = (F; n; K_1; x; c)
  - 由MPFSS的安全性得知K_1 carries no information about μ,
  - $\mathcal{A}$ 要区分$u = a \cdot C_{k,n} + \mu$ 与$u^1 \xleftarrow{R} \mathbb{F}^n$,
  - 在$\mathcal{A}$不知道关于a, μ任何信息的情况下, $\mathcal{A}$ 区分 u 和 u^1等效于打破LPN假设
  - 因此$\mathcal{A}$无法区分$(seed_1, u, v)$和$(seed_1, u^1, v^1)$
  - 归约过程如下:



效率分析

- for G_{prime}: the best known attacks on LPN(k,n,t/n) are the Gaussian elimination attack, which takes time O((1 − t/n)k), the optimal expansion factor is obtained by setting k = t = O(n1/2+ε) for some ε > 0, in which case the Expand algorithm of the VOLE generator expands a seed of size O˜(n1/2+ε) into a pseudorandom VOLE of size O(n);
- for G_{dual}: LPN(n'− n,n',t/n'), seed size:O(t),the Gaussian elimination attack takes time O(1/(1 − t/n')n'−n) ≈ O(e(n'−n)·t /n') when t/n' is sufficiently small, This implies that this approach leads to a VOLE generator with arbitrary expansion factor;

协议应用

基于random VOLE 构造general VOLE

Preprocessing. A trusted dealer picks $(\boldsymbol{r_u}, \boldsymbol{r_v}, r_x) \xleftarrow{R} \mathbb{F}^n \times \mathbb{F}^n \times \mathbb{F}$, sets $\boldsymbol{r_w} \leftarrow \boldsymbol{r_u} r_x + \boldsymbol{r_v}$, and outputs $(\boldsymbol{r_u}, \boldsymbol{r_v})$ to P_0 and $(\boldsymbol{r_w}, r_x)$ to P_1.

Input. P_0 has input $(\boldsymbol{u}, \boldsymbol{v})$, and P_1 has input x.

Protocol. P_1 sends $m_x \leftarrow x - r_x$. P_0 sends $\boldsymbol{m_u} \leftarrow \boldsymbol{u} - \boldsymbol{r_u}$ and $\boldsymbol{m_v} \leftarrow m_x \boldsymbol{r_u} + \boldsymbol{v} - \boldsymbol{r_v}$. P_1 outputs $\boldsymbol{w} \leftarrow \boldsymbol{m_u} x + \boldsymbol{m_v} + \boldsymbol{r_w}$.

Correctness: $\boldsymbol{w} = \boldsymbol{m_u} x + \boldsymbol{m_v} + \boldsymbol{r_w} = (\boldsymbol{u} - \boldsymbol{r_u})x + (x - r_x)\boldsymbol{r_u} + \boldsymbol{v} - \boldsymbol{r_v} + \boldsymbol{r_u} r_x + \boldsymbol{r_v} = \boldsymbol{u}x + \boldsymbol{v}$. Security is straightforward.

We now consider a modification of the above protocol that replaces the ideal random VOLE correlation by the output of the VOLE generator:

Preprocessing. A trusted dealer picks $r_x \xleftarrow{R} \mathbb{F}$, proceeds to compute $(\mathsf{seed}_0, \mathsf{seed}_1) \xleftarrow{R} \mathsf{Setup}(1^\lambda, \mathbb{F}, n, r_x)$, and outputs seed_0 to P_0 and (r_x, seed_1) to P_1.

Offline Expansion. P_0 computes $(\boldsymbol{r_u}, \boldsymbol{r_v}) \leftarrow \mathsf{Expand}(0, \mathsf{seed}_0)$ and P_1 computes $\boldsymbol{r_w} \leftarrow \mathsf{Expand}(1, \mathsf{seed}_1)$.

Input. P_0 has input $(\boldsymbol{u}, \boldsymbol{v})$, and P_1 has input x.

Protocol Π_{VOLE}. P_1 sends $m_x \leftarrow x - r_x$. P_0 sends $\boldsymbol{m_u} \leftarrow \boldsymbol{u} - \boldsymbol{r_u}$ and $\boldsymbol{m_v} \leftarrow m_x \boldsymbol{r_u} + \boldsymbol{v} - \boldsymbol{r_v}$. P_1 outputs $\boldsymbol{w} \leftarrow \boldsymbol{m_u} x + \boldsymbol{m_v} + \boldsymbol{r_w}$.

- 安全性证明

  Proposition 10. *Assuming* $(\mathsf{Setup}, \mathsf{Expand})$ *is a secure* VOLE *generator (as in Definition 5), the protocol* Π_{VOLE} *is a secure vector-*OLE *protocol in the preprocessing model.*

  证明intuition:

  If there exists a simulator can simulate a party's view only by the party's input and output, and the view is indistinguished from the view from the protocol honestly execution.因为这说明协议双方运行协议，获得的消息仅仅源自自己输入以及输出中派生的信息，因此 P_0 不能得到关于 P_1 的私有信息。

  Definition 4.1 *Let* $f = (f_1, f_2)$ *be a functionality. We say that* π securely computes f in the presence of static semi-honest adversaries *if there exist probabilistic polynomial-time algorithms* $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *such that*

  $$\{(\mathcal{S}_1(1^n, x, f_1(x,y)), f(x,y))\}_{x,y,n} \stackrel{c}{\equiv} \{(\mathsf{view}_1^\pi(x,y,n), \mathsf{output}^\pi(x,y,n))\}_{x,y,n}, \text{ and}$$
  $$\{(\mathcal{S}_2(1^n, y, f_2(x,y)), f(x,y))\}_{x,y,n} \stackrel{c}{\equiv} \{(\mathsf{view}_2^\pi(x,y,n), \mathsf{output}^\pi(x,y,n))\}_{x,y,n},$$

  where $x, y \in \{0,1\}^*$ *such that* $|x| = |y|$, *and* $n \in \mathbb{N}$.

  其中，还要考虑双方 joint output distribution。

- 证明过程：

证明思路是：① assume P_0 is corrupt

sence 1

Simulator 通过与可信第三方交互获得的能力

扮演 honest party P_1 与现实中的 adversary P_0 进行交互，

模拟 P_0 的 $view_1$

sence 2.

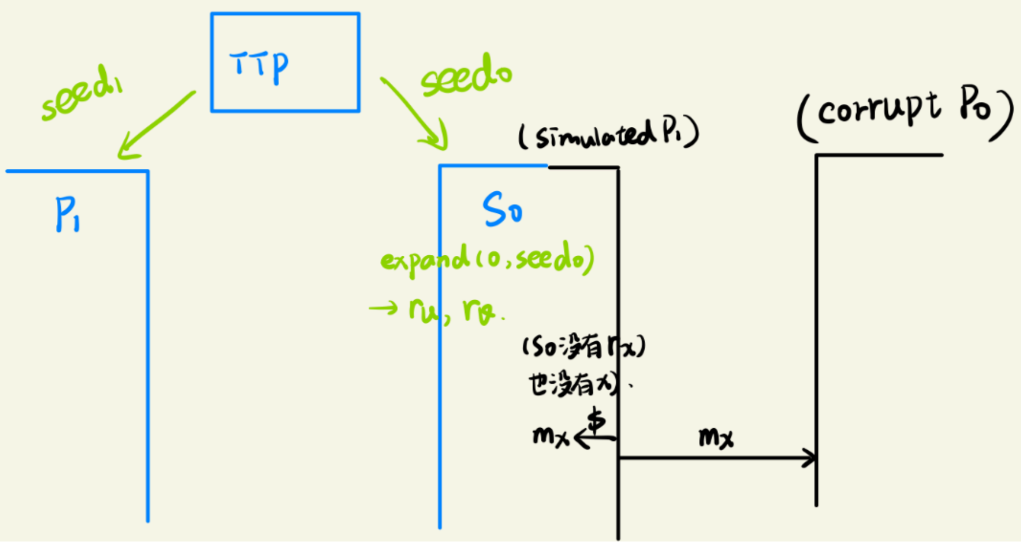
real 中 adversary P_0 与 honest party P_1 进行交互所生成的 $view_2$

$View_1$ 与 $view_2$ 难以区分，则说明该方案在 P_0 is corrupt 下仍是安全的。

归约思路：将 D 区分 $view_1$ 与 $view_2$ 归约到 D 打破 VOLE generator 对应的安全性质。

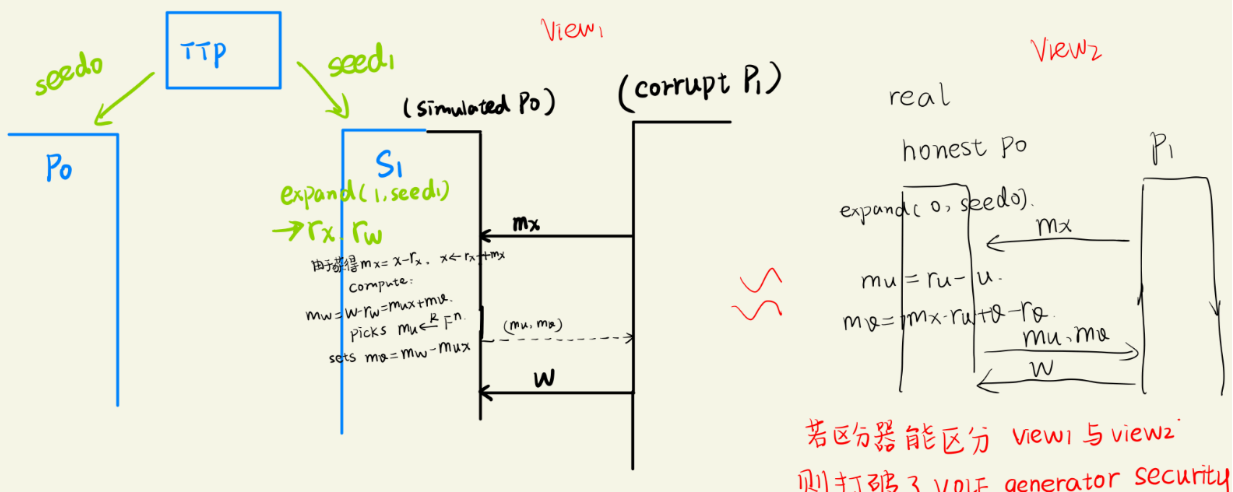
① P_0 is corrupt. 构造 ideal 中的 Simulator S_0，

S_0 在 ideal 中扮演 P_0 角色，S_0 扮演 P_1 与 corrupt P_0 进行交互生成 simulated view.



② P_1 is corrupt. 构造 ideal 中的 Simulator S_1，

S_1 在 ideal 中扮演 P_1 角色，S_1 扮演 P_0 与 corrupt P_1 进行交互生成 simulated view.



$w = m_u x + m_\alpha + r_w$

若区分器能区分 $view_1$ 与 $view_2$

则打破了 VOLE generator security

MPC安全性定义：对于真实世界敌手A，均存在理想世界中的敌手（模拟器）S，使得两个世界的联合分布不可区分；
==》存在模拟器S，能利用理想世界中的能力，模拟的分布与现实世界诚实运行的分布不可区分；