**修改1，生成顺序有小改动，同时为简化，赌场投票改名v1，一共n张票。**

**修改2，验证部分还是全面采用链上decrypt，这是几乎唯一的解决方案。**

**生成r：**

官方列表t=n/2+1

因此 2t = n+2

n=2t-2

其他公众 p 的上限为 p<= n-t = 2t-2 -t = t-2;

所以，p<=t-2

p\_now=0;

t\_now=0;

赌场先行，因为它必须至少控制一票，c（赌场也是其他公众，所以 p\_now++）

开始接受其他选票，如果来自公众的选票 v：p\_now++;

如果来自官方列表的选票 v（可以通过地址判断）：t\_now++;

p\_now 的上限为 t-2，达到后，不再接受来自公众的v

t\_now 的上限为 t-1，达到后，将只再接受官方列表中的一张v，作为最后一张v。

我们需要确保第一张选票来自赌场，以避免竞争条件并确保赌场至少有一票。

我们需要确保最后一张选票来自官方列表，以避免赌场的操纵，因为它拥有私钥。

赌场先选，赌场选择一个v1，用n计算enc(v1)，提交给智能合约，储存enc(v1)

玩家后选，每一个贡献者选择自己的vi，用n计算enc(vi)，提交给智能合约.

最终一共有n个v：分别是v1, v2，v3, …vn。储存enc(v1), enc(v2)… enc(vn)

智能合约用加法同态，计算v\_esum=add( add(enc(v1),enc(v2)), enc(v3)), 播报v\_esum

将结果交给赌场，赌场链下解密dec(v\_esum)得到r\_seed=v1+v2+v3+…+vn，

同时，计算**r = hash\_to\_int( hash(v1+v2+v3..vn) ),** 即**r=hash\_to\_int( hash(r\_seed) )，自己保存r**

**最后，** 计算**enc(r),将其**上传上链

根据题目要求，为奖励这n位用户，但他们花出的gas fee其实很少，我们便将送出0.1% of the total deposit / n 给每个用户

验证r：

1,当一天结束后，赌场将公开v1\_end, v2\_end, v3\_end,.. vn\_end, r \_end 和 **密钥 p, q**

因为储存了enc(v1) enc(v2)…to enc(vn)

验证是否v1\_end==dec(enc(v1)), v2\_end==enc(v2)…

2, 确定v1,v2…的真实性后，可直接得到r\_seed= v1+v2+v3+…vn

再计算得出**r\_v=hash\_to\_int( hash(r\_seed) )，验证 (赌场结束后提供的r): r\_end== r\_v，验证r\_end的真实性。**

dec(enc(r)) 得出r，对比r\_end，确定链上用的r就是同一个r。

**生成x:**

根据上文已经获得enc(r)

用户存下deposit， msg.value=0.01eth，选择随机数k，提交给智能合约, 智能合约保存k, 同时保存k.timestamp，智能合约使用Paillier，用公钥n加密它，得到Enc(k)。

智能合约基于enc(r)和enc(k)，利用Paillier加密方案的同态相加性质计算kr\_esum=add(enc(k),enc(r))，播报kr\_esum 和k.timestamp。

将结果交给赌场，赌场链下解密dec(add(enc(k),enc(r)))得到k+r，再计算x=hash\_to\_int(hash(k+r+k.timestamp)), 最后上传：enc(x) 和x的奇偶性给智能合约。

智能合约根据x的奇偶性确定胜负（偶数bettor赢，奇数赌场赢），并相应地支付用户。

验证x：

从前面步骤我们已经验证了r的真实性，k是自动保存的用户自己公开的自选随机数, k.timestamp则是自动保存的时间戳。dec 用到的也是前面提过的p和q

基于此，我们可以在利用链上储存的enc(x)，计算dec(enc(x))得到x。

同时，我们可以链上验算 hash\_to\_int(hash(k+r+k.timestamp))，与x对比，由此证明x的有效性

确认x有效后，我们检查x 的奇偶性 是否与 输赢对应记录的奇偶性相同。

最后，任何验算不通过即是赌场作弊，双倍赔钱（赔0.02eth）。

所有验算自动完成，无需用户输入input，避免讹钱。

k.timestamp 避免了有人可能用同一个k重复赢钱的问题。