# 零知识证明 - Knowledge-of-Exponent Assumption 介绍

同态加密我们知道有一个应用场景就是需要对数据进行计算密集型的计算，比如做人工神经网络的训练，我们希望第三方给我们做训练，但是我们不想暴露原始数据，就可以用同态的加密来做。但是，这里有一个问题，对方是不是真的在用我们提供的数据在训练？在零知识证明中也有这个问题，prover 是不是用 verifier 提供的数据在计算，还是在凑数据骗过验证。

我们先来看这样的一个协议：

> **Alice**：选择一个随机数 k， 计算 $a' = a^k(modn)$, 然后把 $(a', a)$ 发送给 Bob
>
> **Bob**： 计算 $b = a^c(modn)$, $b' = a'^c(modn)$, 然后把$(b', b)$ 发送给Alice
>
> **Alice**： 验证：$b^k = b'(modn)$, 如果通过了，那说明，Bob确实用了a 和 a′ 在计算。

我们知道，通过 $(a', a)$ 是无法求出k的，这就是离散对数问题。Bob在无法求出k的前提下 肯定无法构造出 $b^k = b'(modn)$, 唯一的方法就是老老实实用 $(a, a')$ 做运算。

我们知道，零知识证明中，使用了多项式，一般verifier 会给出一个 随机数s，让prover造证明，但是prover 不一定用这个随机数来计算，为了保证prover 用这个来计算，我们就可以用上面的方法来实现。

协议如下：

> g 是椭圆曲线的生成元，下面的公式全部用乘法群的方式来表示(对ecc来说，指数运算对应乘法，乘法对应加法)。
>
> **verifier**： 随机生成 s , k 两个随机数，计算 $g^s$ 和 $(g^s)^k$, 然后把 $g^s$ 和 $g^{sk}$ 发送给 prover
>
> **prover**: 计算 $(g^s)^c$ 其中 c 是多项式的系数，$(g^{sk})^c$, 然后把 $g^{sc}$ 和 $g^{skc}$ 发送给 verifier
>
> **verifier**： 验证 $(g^{sc})^k = g^{skc}$

上面的协议我们只是求出多项式的一项，如果是完整的多项式，会稍微复杂一点，我们就用一个例子来表示

$$p(x) = x^3 - 3x^2 + 2x$$

协议如下：

> **verifier**:
>
> 随机生成 s，k 两个随机数，并且计算$g^{s^3}$,$g^{s^2}$,$g^s$, 并且计算 $g^{ks^3}$,$g^{ks^2}$,$g^{ks}$
>
> **prover**:
>
> $g^p = g^{p(s)} = (g^{s^3})^1 . (g^{s^2})^{-3} . (g^s)^2 = g^{s^3-3s^2+2s}$
>
> $g^{p'} = g^{kp(s)} = g^{k(s^3-3s^2+2s)}$
>
> 发送 $g^p, g^{p'}$ 给verifier
>
> **verifier**:

> 验证: $(g^p)^k = g^{p'}$

显然，这个公式是成立的，而且，如果prover 不采用用户提供的数据来计算，因为无法计算出k的值，要刚好凑出两个数满足 $(g^p)^k = g^{p'}$ 是一个概率可以忽略的事件，特别是问题规模足够大的情况下。