加密密钥产生的过程(*ntru_crypto_ntru_encrypt_keygen*)

1. 初始化三个值 dF_1, dF_2, dF_3 （其实是预先给定的）分别代表三个系数在 $\{-1, 0, 1\}$ 之间的 $(Z/qZ)[X]/(X^N-1)$ 意义下的多项式 f_1, f_2, f_3 中 1、-1 的个数
2. 利用 $drbg$ 产生 -1、1 在多项式中的索引，由此初始化多项式 f_1, f_2, f_3 并计算 $F = f_1 * f_2 + f_3$
3. 计算 $f = 1 + pF$
4. 计算 f_q^{-1} :
    - 首先计算 f_2^{-1}
    - 再计算 f_q^{-1} (其中 $q = 2^t$)
5. 初始化 d_g （其实是预先给定的） 代表多项式 g 中 $\{1, -1\}$ 的个数
6. 与上述 2 相同方法产生多项式 g
7. 计算 $h = p * (f_q^{-1} * g)$

需要注意的地方：

- 代码中多次用位运算代替取模运算，即 $a\&(b-1) = a\%b \quad if\ b = 2^k$
- 代码中交换 a, b 用到了三次异或操作：

```
a ^= b; b ^= a; a^= b;
```

- 一些需要关注的函数及其作用：
    - ***ntru_gen_poly*** 产生多项式（系数的索引）
    - ***ntru_ring_mult_indices*** 计算多项式 a, b 的乘法（其中多项式 b 的系数属于$-1, 0, 1$ ）
    - ***ntru_ring_mult_product_indices*** 计算多项式 a, b 的乘法（其中多项式 $b = b_1 * b_2 + b_3$）并且 $b_i \quad i \in [1, 3]$ 的系数属于 $-1, 0, 1$ ）
    - ***ntru_ring_inv*** 计算多项式 $mod\ 2$ 意义下的逆多项式，此处用到了扩展的欧几里得算法，具体可参见 Extended Euclidean algorithm
    - ***ntru_ring_lift_inv_pow2_product*** 计算多项式 $mod\ q\ (q = 2^k)$ 意义下的逆多项式
- 代码中将密钥存储在 $keyblob$ 中的过程应该可以省略

加密过程(*ntru_crypto_ntru_encrypt*)

1. 根据 dr 生成多项式 r
2. 计算多项式 $R = h * r$ （ h 为公钥 ）
3. 后面对多项式 R 和对明文多项式 m 的预处理操作（可参见论文）
4. 计算密文 $e = R + m'$

解密过程(*ntru_crypto_ntru_decrypt*)

1. 计算 $F * e$ （其中 $F = f_1 * f_2 + f_3$ 为私钥 ）
2. 计算多项式 $A = e(1 + pF) = e + epF \ \ modq$
3. 将 A 的系数中心化，使其系数的范围在 $[-q/2, q/2)$
4. $a = A = m'(1 + pF) = cm' \ \ modp$
5. 根据 cm' 解出 m （可参见论文）