**Solution Description**

Our implementation follows standard cryptographic primitives, whose security guarantee has been analyzed and verified for decades. We give some brief description below.

**1.** We use the well-studied Ed25519 curve. Let us denote to be the base point, whose order is a large prime .

**2.** We use a homomorphic variant of the standard ElGamal encryption scheme.

**Key Generation:**

* The private key is random number . (Used in Ed25519)
* The public key is .

**Encryption:**

Given a 40-bit message , the encryption is:

where is a random number sampled from range . Therefore, the ciphertext of is a pair .

**Decryption:**

We have

Therefore, we can solve this discrete log problem when is small. When is large (40 bits), we can solve it with the baby-step-giant-step algorithm by precomputing a lookup table. We discuss this later.

**3. Security.** Because our scheme is basically the standard ElGamal encryption on Elliptic Curve, the security guarantee is based on the hardness assumption of the discrete logarithm problem: given public key , and any ciphertext , it is difficult to compute the corresponding discrete log on base .

According to the NIST recommendation (<https://www.keylength.com/en/4/>) and the analysis of the Curve25519 paper (“Curve25519: new Diffie-Hellman speed records”), the scheme provides minimum 128-bit security.

**4. Baby-Step-Giant-Step** to speedup decryption. To facilitate the decryption, we can break any plaintext as follows:

,

where , . Here, is the number of bits for the baby step, while is the number of bits for the giant step. We can precompute all giant steps and store them in a lookup table. In this way, the decryption phase only needs baby steps to recover the plaintext.

In our submission, we use , and the giant-step lookup table is also submitted (~1GB). **NOTE: this table is fixed information for the curve Ed25519 and can be shared publicly.**