1. System Parameters
The system uses a group G of large prime order q. In our implementation, we use a subgroup of Z_p^* where $p = 2q+1$.

- **Public Key:** (g_1, g_2, c, d, h)
- **Private Key:** (x_1, x_2, y_1, y_2, z)
- **Hash Function:** A universal one-way hash function H (we will use SHA-256).

2. Key Generation

- Generate a large safe prime p.
- Select two random generators g_1, g_2 in Z_p^*.
- Select random secret scalars x_1, x_2, y_1, y_2, z from $\{0, \dots, p-2\}$.
- Compute the public values:
  - $c = g_1$^$x_1 * g_2$^$x_2 \ (mod\ p)$
  - $d = g_1$^$y_1 * g_2$^$y_2 \ (mod\ p)$
  - $h = g_1$^$z \ (mod\ p)$

3. Encryption
To encrypt a message m (converted to an integer):

- Select a random r.
- Compute:
  - $u_1 = g_1$^$r \ (mod\ p)$
  - $u_2 = g_2$^$r \ (mod\ p)$
  - $e = h$^$r * m \ (mod\ p)$
- Compute the hash $alpha = H(u_1, u_2, e)$.
- Compute the verification tag $v = c$^$r * d$^$r*alpha \ (mod\ p)$.
- **Ciphertext:** (u_1, u_2, e, v).

4. Decryption & Validation
Given ciphertext (u_1, u_2, e, v):

- Compute $alpha = H(u_1, u_2, e)$.
- **Validation Step:** Verify if u_1^$(x_1 + y_1*alpha) * u_2$^$(x_2 + y_2*alpha) \equiv v \ (mod\ p)$.
  - If this equality does *not* hold, the ciphertext is invalid (reject it).
- If valid, recover the message: m = e * (u_1^z)^(-1) (mod p).