DLP to build a fixed length collision resistant hash function

Assumptions: Used previously built PRG to generate h.

Input format:

1. It will ask for a safe prime number p in integer format.
2. It will ask for a generator (primitive root) of that prime.
3. It will then ask for a seed to generate h (b/w 1 to prime) via PRG.
4. Then, it'll ask to input a number x1 ranging from 0 to (prime-1).
5. Then, it'll ask to input a number x2 ranging from 0 to (prime-1).

Output Format:

1. It will output the DLP based fixed length hash in binary format.

```
PS C:\Users\Sudipta Halder\Desktop\IIITH ASSIGNMENTS\POIS> python .\6_dlp_collision_resistant_hash.py
*********************************************************************************************************
The logic is as follows
The length of prime numebr(in binary) should be equal to x1, x2(in binary)(The data which will be provided)
So ideally we should know data length from user and then choose a prime of that length
In case of merkle damgard transform, the two datas(msg and vector) will be of same length say n
So, then we should choose a prime number which is of length n in binary
Then choose primitive root or generator g
Then choose h randomly b/w 1 to (prime-1)
Then take input of data x1 and x2, in range 0 to (prime-1)
Or we can choose prime beforehand, tell user that our prime is of n bits
So, data will be divided into blocks of size n(length of prime)
So, if your data is of less than n bits in any block, pad 0's at end
*********************************************************************************************************

Enter the prime number(The prime should be such that p-1/2 should also be prime. Sophie Germain Prime)(1907): 1907
Enter the generator(Primitive root for the prime)(987): 31
Input a seed in binary for PRG: 10011001101101

Randomly selected h from 1 to prime: 1516

Enter a number b/w 0 to 1906(11101110011) in binary(keep number of bits(11) same as prime for length halving): 10110001101

Enter a number b/w 0 to 1906(11101110011) in binary(keep number of bits(11) same as prime for length halving): 10010111100
x1: 1421
x2: 1212
Length of (x1+x2): 22
x1_mod_p: 1421, x2_mod_p: 1212
Hash Res: 10100011100
Hash Res length: 11
```

Working Flow:

1. At first, h will be calculated using PRG which will be in range [1, prime].
2. Upon getting all inputs, the function ` calculate_dlp_hash(prime, generator, h, x1_int, x2_int)` will be called.
3. It will first check whether x1 and x2 are in range of (0, prime-1) or not. If not, then it will apply mod operation on then to bring them in range (0, prime-1).
4. Then it calculate (g^{x1} mod p * h^{x2} mod p)mod p where p is prime. This result will then be output as binary.
5. This function ensures 50% data compression when data x1 and x2 are of same length and length = prime length in binary format i.e., if x1 and x2 is of length n, then hash result will output a n-bit hash.

Provable compression functions

Choose a random 2000-bit prime p and random $1 \le u, v \le p$.

For $m,h \in \{0,\ldots,p-1\}$ define $h(H,m) = u^H \cdot v^m \pmod p$

Fact: finding collision for h(.,.) is as hard as solving "discrete-log" modulo p.

A Fixed Length Hash Function

Let P be a polynomial time algorithm that on input 1^n outputs a cyclic group G of order q (length of q is n) and generator g

Gen: Run $P(1^n)$ obtain (G,q,g); select uniformly at random an element h from G; Output **s = (G,q,g,h)**

H: On input x_1 and x_2 (both in the range 0 to q-1), output

$$H^s(x_1,x_2) = g^{x_1} h^{x_2}$$

Theorem: If discrete logarithm problem is hard then the above is a fixed length collision resistant hash function