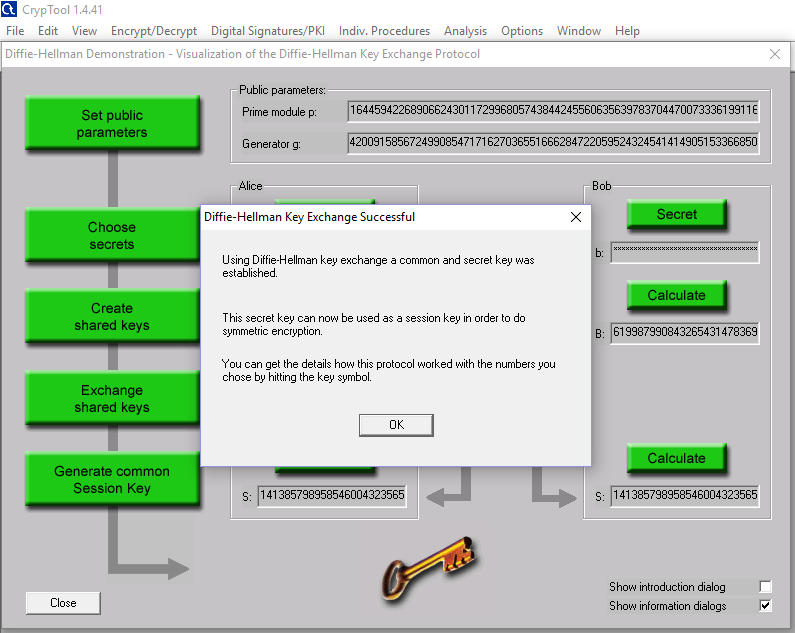
1c.)

1. Alice’s secret key: **78 decimals** 101965792666427631757328750786082934471278993060544181333928909597605975664320, Alice’s shared key: **76 decimals** 3005264838784582385644764543046566512268625950382486498025509433356806871669, Bob’s secret key: **78 decimals** 131335006805181961356210025915806252972812277088697275781674237337656463245935, Bob’s shared key: **77 decimals** 61998799084326543147836939205351392868219179125456037064548702040189976069758
2. Alice’s secret key has 78 decimals, so its binary number is 1001110(**7 binary digits**) Alice’s shared key: has 76 decimals, so its binary number is 1001100(**7 binary digits**), Bob’s secret key has 78 decimals, so its binary number is 1001110(**7 binary digits**), Bob’s shared key has 77 decimals, so its binary number is 1001101(**7 binary digits**)
3. **p:** 164459422689066243011729968057438442455606356397837044700733361991165383163799 (decimal is 78, so binary is 1001110->**7 binary digits**)

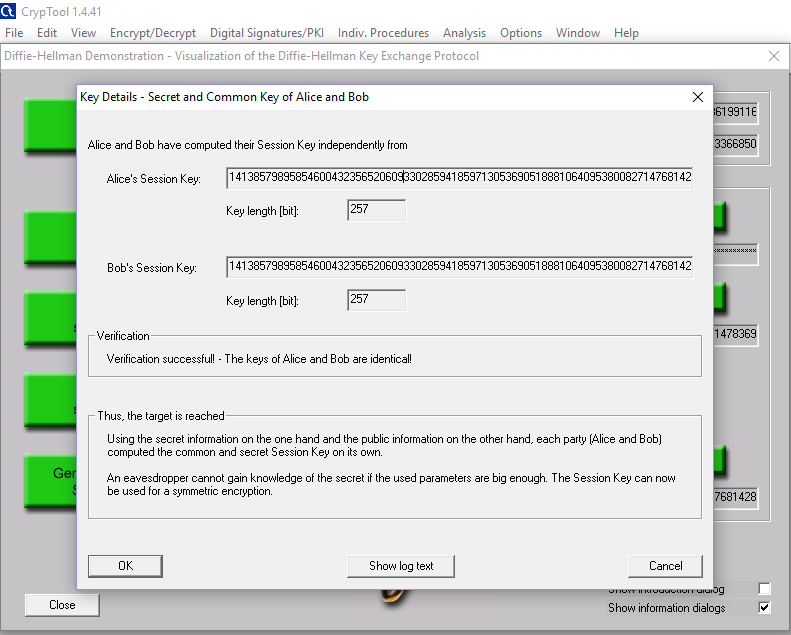
**g:** 156290504842009158567249908547171627036551666284722059524324541414905153366850(decimal is 78, so binary is 1001110->**7 binary digits**)

1d.)

1. **No**
2. **Yes**



1. 141385798958546004323565206093302859418597130536905188810640953800827147681428 (**78 decimal digits long**)

1e.) ****

1f.) **Yes, the common session key between Alice and Bob match**

2c.) Alice’s shared key: g^a mod p, Bob’s shared key: g^b mod p

13^21 mod 67 = 8 and 13^12 mod 67 = 25



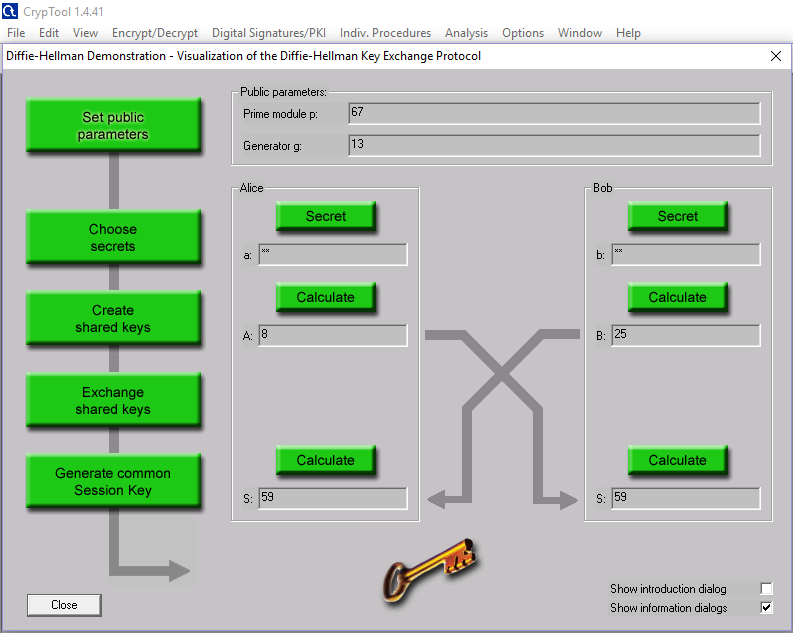
2d**.) Alice’s shared key: 8, Bob’s shared key: 25**

2e.) A=B^a mod p and B=A^b mod p

A= 25^21 mod 67=59 and B= 8^12 mod 67 = 59



2f.) Alice’s shared key is 8 and Bob’s shared key is 25, therefore they do not match. However Alice and Bob’s session keys match with an S value of 59.



3.) **The prime for the 1536 bit group is: 2^1536 - 2^1472 - 1 + 2^64 \* {[2^1406 pi] + 741804}**

* **The generator (g value) is: 2**

4.) a: Briefly describe what vulnerability in the DH key exchange does this attack exploit.

* **The main vulnerability with DH includes: allowing the attacker to compute a shared key. For example, if you are given: y=g^a mod p to compute Alice’s secret “a” this would allow the attacker to compute the shared secret the same way that Alice did.**

b: Is this vulnerability something to be concerned about? List some protocols used in the real world that use Diffie Hellman key exchange. Are these vulnerable to LogJam?

* **Yes, this vulnerability is something to be concerned about. The Transport Layer Security (TLS) protocol uses Diffie Hellman key exchange. TLS is vulnerable to LogJam because LogJam allows a man-in-the-middle-attack. It allows the attacker to read and modify any data passed over the connection. The TLS protocol has a flaw in the protocol rather than an implementation vulnerability resulting in this LogJam success**
* **Also, most Secure Socket Layer (SSL) servers use the same modulus as the Diffie Hellman key exchange, because using the one provided with the SSL library is easier than generating your own. Reusing the same modulus as everybody else is not a big issue; DH tolerates that just fine. However, it means that if an attacker invests a lot of computations in breaking one DH instance that uses a given modulus p, the same attacker can reuse almost all of the work for breaking other instances that use the same modulus p.**