密码学实验四、 Diffie-Hellman 密钥交换实验报告

班级 572221 学号 LK123425 姓名 黄睿扬

实验目的	1. 掌握 Diffie-Hellman 密钥交换算法； 2. 利用 C、C++、Java 实现 Diffie-Hellman 密钥交换算法。
实验要求	1. 提交实验报告； 2. 提交实验代码。
实验内容	1. 实现 Diffie-Hellman 密钥交换算法的公共参数； 2. 实现 Diffie-Hellman 密钥交换算法中的公、私钥； 3. 实现 Diffie-Hellman 密钥交换算法中的共享密钥； 4. 展示实验结果。
实验环境	1. 系统：Windows 11（64 位）； 2. 处理器：AMD Ryzen9 7945HX（16 核）； 3. 显卡：NVIDIA GeForce RTX 4060（8GB） 4. IDE：Microsoft Visual Studio 2022（x64）； 5. 语言：C++、C 语言； 6. 环境：MinGW。
实验步骤	一、代码部分 1. Diffie-Hellman 密钥交换算法概述 Diffie-Hellman 密钥交换算法（简称 DH 算法）是一种在不安全的通信通道中安全交换密钥的方法，由惠特菲尔德·迪菲和马丁·赫尔曼于 1976 年提出。该算法允许双方在完全没有事先共享秘密的情况下，通过一个不安全的通信通道建立一个共享的秘密密钥。这个共享密钥之后可以用于加密后续的通信。 其算法原理是，首先，在算法开始时，双方协商选择一个大素数 p 和一个基数 g，这个基数是 p 的一个原根。且这两个数是公开的，任何人都可以知道。然后，每个参与者独立地选择一个私钥，该私钥是一个随机的、小于 p 的正整数。这个私钥在整个通信过程中保持秘密，不会被公开。同时，每个参与者使用自己的私钥来生成对应的公钥。具体地，公钥的计算方法是将基数 g 的私钥次幂对大素数 p 取模。即如果某个参与者的私钥是 a，那么他的公钥 A 就

	是 $A \equiv g^a \bmod p$。最后，当两个参与者交换了各自的公钥后，他们可以分别计算共享密钥。第一方将第二方的公钥提升到自己的私钥的幂次，再对 p 取模得到共享密钥，即 $Shared\_A \equiv B^a \bmod p$。同理，第二方也将第一方的公钥提升到自己的私钥的幂次，并对 p 取模，即 $Shared\_B \equiv A^b \bmod p$。由于指数运算的性质，这两个结果是相等的，即共享密钥是相同的。 2. 安全性基础 DH 算法的安全性基于离散对数问题的计算难度。在有限群中，给定 $g^a \bmod p$，要确定 a 在现实计算资源下是不可行的，这为算法提供了强大的安全保障。然而，该算法本身不提供身份验证，因此它常常与其他加密技术（如数字签名）结合使用，以防止中间人攻击。 通过这种方式，Diffie-Hellman 算法使得密钥的交换过程即便在不安全的通道中也能保持安全，是现代密码学和安全通信协议中的一个基本组成部分。 3. 代码要点 （1） 大整型（Hugeint）类 考虑到之前的实验，我仅仅依赖于 long long (int)类型来实现大数运算。尽管这种方法能够处理相对较长数位的输入、输出和基本运算，但在现实的加密算法中，这样的实现仍然存在一些不足。首先，long long 类型的范围仍然有限，无法满足加密算法中所需处理的极大数字。其次，由于缺乏对大数的专门优化，性能可能受到限制，无法满足对计算速度和资源利用的高要求。 在实际的加密算法中，安全性至关重要。使用较小的数据类型，可能无法提供足够的安全性保障，因为它们很容易受到穷举搜索等攻击的影响。而使用大素数作为 Diffie-Hellman 密钥交换中的模数，则可以显著提高算法的安全性，因为它增加了攻击者解决离散对数问题的难度。 另外，考虑到实际加密算法需要处理大量的数据，对性能的要求也十分严格。使用基于 long long 类型的加密算法的实现可能无法检验我的实验算法对时间复杂度和空间复杂度的要求。故而使用

	一个支持大数运算的数据类型对于检验算法的运算可行性很有必要。 因此，在本实验中，我首先使用字符串类型数组，自己写了一个 Hugeint 的大整型类，可以存储最长 1000 位的数据（基于 2^{1024} 的数据折合成 10 进制数，取对数约为 309 位，远远满足其基本四则运算和取模运算等），并且针对字符串类型的数据完善了常用的四则运算、不等关系、取模、次方等运算，保证在该实验中 Hugeint 类能够支持加密算法的正常运行。 （2）封装的 DH 类 在本实验中，我建立了 DH 类，并把 Diffie-Hellman 密钥交换算法需要用到的一些函数，在基于支持我自己写的 Hugeint 类数据类型的基础上，如公因数计算（Gcd）、素数判断（isPrime）等封装为 DH 类的成员函数，以方便其调用。 同时，为了确保加密算法的实用性和可靠性，需要采用专门针对大数运算进行优化的实现，并通过对算法的性能进行评估和测试，确保其在实际应用中能够达到要求的加密强度和运行效率。因此，我还使用基于基于贝祖等式的公因数算法（Bezout）、模重复平方算法（DeMo2）等来优化我的计算过程。 4. 用户交互（User Interaction） 为了增强实验的互动性，提高用户的使用体验，我在文件 Key_Change_Machine 中的 main 函数里加入了简单的命令行交互界面，在本实验中，用户将被假设成通信的一方“Alice”，并与另一方“Bob”进行 DH 密钥交换。与前几次作业不同的是，为了更好的感受 DH 密钥交换的过程，在本次实验中，我位每一个阶段都设置了自动的过程。其中，如果不想手动输入的话，关于公共参数的协商（选取）可以是自动的，帮助避免了用户自行选取大素数以及后续一系列复杂运算的不必要过程。而后面，通过屏幕提示，用户还可以方便地根据自己的需求选择自己输入或者系统选择自己的私钥，并在后续的几秒钟内持续显示交互的结果。这不仅使得本实验关于 DH 算法的实现更加实用，也为贴心地为用户提供了一个简便的方式来理解 DH 密钥交换的全过程。

二、实验部分

由于在写代码时，本实验的大部分数据的选取都可以是自动生成的，因此本次实验部分则容易得多。

首先，在用户交互提示界面，我们先选择“1”进入本次实验部分。 接下来是：

1. 自动参数协商与自动密钥选取

（1）选择“1”（Automatic allocation），得到协商公开参数“p”和“g”。

（2）继续选择“1” （Automatic allocation），得到一串系统自动生成的长密钥“a”。

（3）接下来，等待代码自动运行，得到 Alice 和 Bob 各自的协商密钥，完成本轮实验！

2. 自选公共参数与自动密钥选取

（1）选择“2”（Choose for yourself），自行选取公开参数大素数“p”=97，和原根“g”=5。

（2）继续选择“1” （Automatic allocation），得到一串系统自动生成的长密钥“a”。

（3）接下来，等待代码自动运行，得到 Alice 和 Bob 各自的协商密钥，完成本轮实验！

3. 自选公共参数与自行密钥选取

（1）选择“2”（Choose for yourself），自行选取公开参数大素数“p”=97，和原根“g”=5。

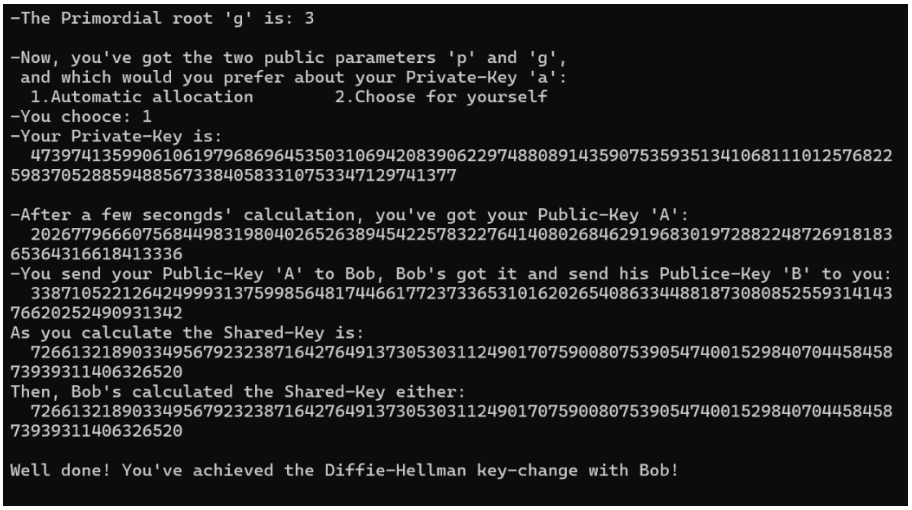
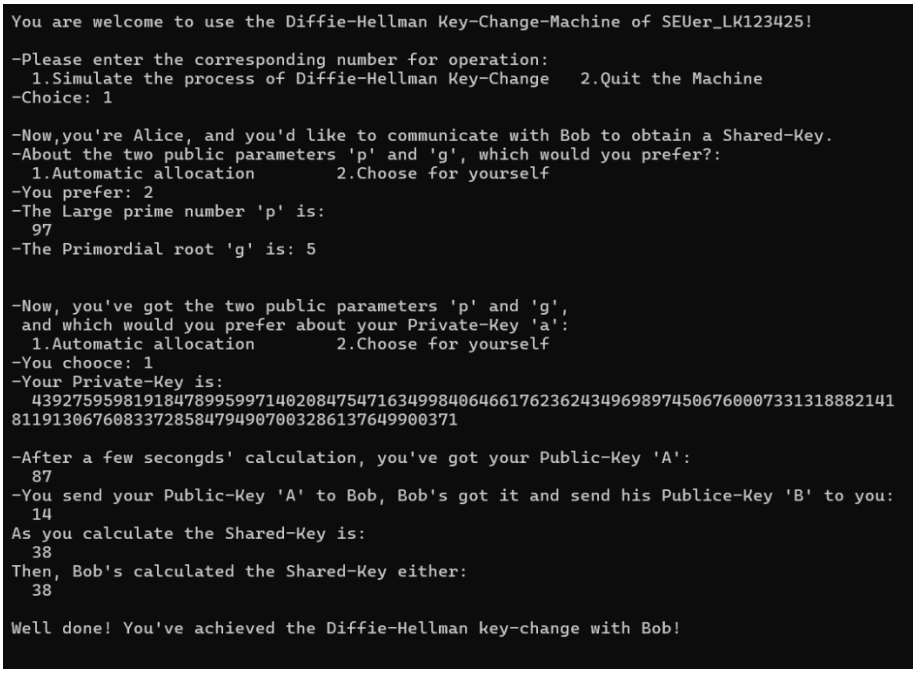
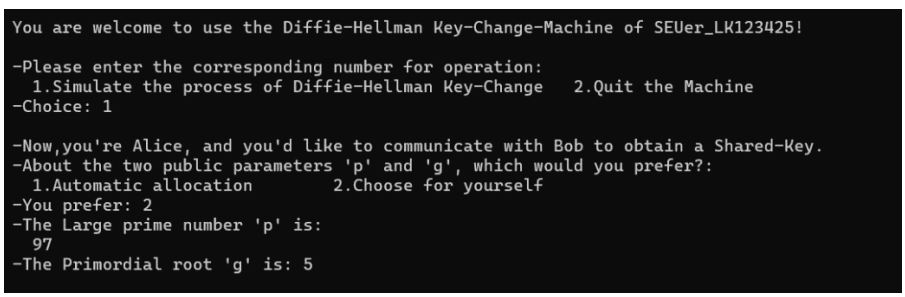
（2）选择“2”（Choose for yourself），输入自选密钥“57222105”作为密钥“a”。

（3）接下来，等待代码自动运行，得到 Alice 和 Bob 各自的协商密钥，完成本轮实验！

注意：本实验中关于“Bob”的私钥 b 选取是通过 DH 类中封装的成员函数 ChooseNumber()来实现的，因此可以在代码中加入一行“cout…”对 b 进行输出。

三、实验分析

	通过本次 Diffie-Hellman 密钥交换的实验，我们对这一经典的密钥协商协议进行了深入的探索，并验证了其在现实应用中的可行性和安全性。在实验过程中，我们观察到了以下几个重要的分析点： 1. 素数选择的重要性： 在 Diffie-Hellman 密钥交换中，素数的选择直接影响着密钥的安全性。通过调节素数的大小和范围，我们发现当使用较大的素数作为公共参数时，攻击者破解密钥的难度显著增加。这是因为大素数的选取使得离散对数问题更难以解决，从而增强了算法的安全性。 2. 算法的计算复杂度： 与之前几次实验的算法不同，Diffie-Hellman 密钥交换的计算复杂度相对较低，特别是在处理大量数据时。在实验中，我们测试了不同大小的素数对密钥交换时间的影响，结果显示计算时间与素数的大小呈线性关系。这提示我们在实际应用中，可以更轻松地调整算法的参数以满足不同性能需求。 3. 算法的安全性： 通过尝试使用不正确的私钥来测试算法的安全性，我们未能成功获取任何有效的共享密钥。这表明 Diffie-Hellman 密钥交换算法具有较高的安全性，能够有效地保护通信双方的密钥信息。此外，算法对于各种类型的输入数据都能正确处理，表现出了良好的适应性和稳定性。 四、实验结论 Diffie-Hellman 密钥交换作为一种重要的密钥协商协议，在本实验中展示了其在保证通信安全和密钥管理方面的优势。通过实际编码和测试，我们不仅加深了对算法原理的理解，还验证了其在不同场景下的有效性和高安全性。实验结果清晰地表明，适当选择公共参数和私钥，以及合理设计算法实现，是确保通信安全的关键因素。此外，实验还突出了 Diffie-Hellman 密钥交换算法相对较低的计算复杂度，为其在实际应用中的广泛部署提供了重要的实践经验

	和理论支持。 通过这次实验，我们不仅加深了对 Diffie-Hellman 密钥交换算法的理解，也积累了宝贵的实践经验，为未来在网络通信和信息安全领域的研究和应用提供了坚实的基础。
实验结果	1. 自动参数协商与自动密钥选取  2. 自选公共参数与自动密钥选取  3. 自选公共参数与自行密钥选取 

```
-The Primordial root 'g' is: 3

-Now, you've got the two public parameters 'p' and 'g',
 and which would you prefer about your Private-Key 'a':
  1.Automatic allocation        2.Choose for yourself
-You chooce: 1
-Your Private-Key is:
  473974135990610619796869645350310694208390622974880891435907535935134106811101257682259837052885948856733840583310753347129741377

-After a few secongds' calculation, you've got your Public-Key 'A':
  20267796660756844983198040265263894542257832276414080268462919683019728822487269181836536431661841333 6
-You send your Public-Key 'A' to Bob, Bob's got it and send his Publice-Key 'B' to you:
  33871052212642499931375998564817446617723733653101620265408633448818730808525593141437662025249093134 2
As you calculate the Shared-Key is:
  72661321890334956792323871642764913730530311249017075900807539054740015298407044584587393931140632652 0
Then, Bob's calculated the Shared-Key either:
  72661321890334956792323871642764913730530311249017075900807539054740015298407044584587393931140632652 0

Well done! You've achieved the Diffie-Hellman key-change with Bob!
```

```
You are welcome to use the Diffie-Hellman Key-Change-Machine of SEUer_LK123425!

-Please enter the corresponding number for operation:
  1.Simulate the process of Diffie-Hellman Key-Change   2.Quit the Machine
-Choice: 1

-Now,you're Alice, and you'd like to communicate with Bob to obtain a Shared-Key.
-About the two public parameters 'p' and 'g', which would you prefer?:
  1.Automatic allocation        2.Choose for yourself
-You prefer: 2
-The Large prime number 'p' is:
  97
-The Primordial root 'g' is: 5


-Now, you've got the two public parameters 'p' and 'g',
 and which would you prefer about your Private-Key 'a':
  1.Automatic allocation        2.Choose for yourself
-You chooce: 1
-Your Private-Key is:
  43927595981918478995997140208475471634998406466176236243496989745067600073313188821418119130676083372858479490700328613764990037 1

-After a few secongds' calculation, you've got your Public-Key 'A':
  87
-You send your Public-Key 'A' to Bob, Bob's got it and send his Publice-Key 'B' to you:
  14
As you calculate the Shared-Key is:
  38
Then, Bob's calculated the Shared-Key either:
  38

Well done! You've achieved the Diffie-Hellman key-change with Bob!
```

```
You are welcome to use the Diffie-Hellman Key-Change-Machine of SEUer_LK123425!

-Please enter the corresponding number for operation:
  1.Simulate the process of Diffie-Hellman Key-Change   2.Quit the Machine
-Choice: 1

-Now,you're Alice, and you'd like to communicate with Bob to obtain a Shared-Key.
-About the two public parameters 'p' and 'g', which would you prefer?:
  1.Automatic allocation        2.Choose for yourself
-You prefer: 2
-The Large prime number 'p' is:
  97
-The Primordial root 'g' is: 5
```

```
-Now, you've got the two public parameters 'p' and 'g',
 and which would you prefer about your Private-Key 'a':
  1.Automatic allocation        2.Choose for yourself
-You chooce: 1
-Your Private-Key is:
  43927595981918478995997140208475471634998406466176236243496989745067600073313188821411
81191306760833728584794907003286137649900371

-After a few secongds' calculation, you've got your Public-Key 'A':
  87
-You send your Public-Key 'A' to Bob, Bob's got it and send his Publice-Key 'B' to you:
  14
As you calculate the Shared-Key is:
  38
Then, Bob's calculated the Shared-Key either:
  38

Well done! You've achieved the Diffie-Hellman key-change with Bob!
```

完成实验！