算法设计与分析课程设计实验报告

黄元通 [1)5)] 于之晟 [2),5)] 李欣宇 [3)5)] 付大川 [4)5)]

[5)]东北大学 计算机科学与工程学院 人工智能 1902 班

[1)]学号：20195063 [2)]学号：20195286 [3)]学号：20195215 [4)]学号：20195184

摘 要 在本次的算法设计与分析课程设计中，针对任务书所给的三个不同场景，主要依据 ElGamal 公开密钥密码体制和 Pedersen 承诺方案，在满足程序设计要求的限定下，小组设计了问题中三个场景的解决方案，并使用 C++语言编写实现了三个场景的模拟展示程序。三个场景下的模拟展示程序不仅完成了基本要求，而且实现了隐私性和安全性，保障了信息安全。

关键词 算法设计与分析；信息加密；ElGamal 公开密钥密码体制；Pedersen 承诺方案；C++；GMP 库

Algorithm Design and Analysis Course Project Experiment Report

Huang Yuantong[1)5)] Yu Zhisheng[2)5)] Li Xinyu[3)5)] Fu Dachuan[4)5)]

[5)]Class 1902 of Artificial Intelligence, Department of Computer Science and Engineering, Northeastern University

[1)]Student Number: 20195063 [2)]Student Number: 20195286 [1)]Student Number: 20195215 [1)]Student Number: 20195184

Abstract

During the period of Algorithm Design and Analysis Course Project class, based on the use of ElGamal Cryptosystem and Pedersen Commitment, the team came up with solutions to each of three problems described in the task requirement. And we coded programs to simulate and demonstrate the process of how our solutions work, which are coded in C++ language and meet all the requirements as well. While the program designed for these three scenarios not only fulfilled the basic requirements, they ensure privacy and security as well, guaranteed security for all the information.

Key words Algorithm Design and Analysis; information encryption; ElGamal Cryptosystem; Pedersen Commitment; C++; GMP library

1 问题形式化定义

所有三个情景的问题形式化定义相同，如下：

输入：

数学表示：S_A、S_B

含义：S_A为 Alice 的成绩，S_B为 Bob 的成绩。

说明：在程序中，双方成绩在数据初始化、及游戏开始前，通过 *Menu()*菜单函数时用户的选择进行对其值进行初始化。其数值满足：

$S_A, S_B \in (A\_Class, B\_Class, C\_Class, D\_Class)$

输出：

数学表示：Yes / No

含义：指 Alice 端和 Bob 端在游戏结束前做出的“对方成绩和自己的是否相同”判断相应的输出界面。Yes 指“判断得对方和自己成绩相同”时的相应输出界面；No 指“判断得对方和自己成绩不同”时的相应输出界面。

2 参数与函数基本库 Basic

三个情景均使用到同一个自定义的参数与函数基本库 Basic（C++实现的类，在三个游戏中均设定 Basic 类的全局对象 *basic*），Basic 中参数与函数如下：

2.1 常量

基本库中的常量如下表 2.1 所示，其中：

1. 第一大类 *p*、*Elgamal_g*、*Pedersen_g*、*Pedersen_h* 为教师提供 txt 文件中的大整数。
2. 第二大类 *A_Class* 等为相应中文字符'优'、'良'、'中'、'差'所对应 Unicode 值。

   $A\_Class = 20248, B\_Class = 33391$

   $C\_Class = 20013, D\_Class = 24046$
3. 第三大类 *seed* 类型为 gmp_randstate_t，是大整数随机数生成函数的种子，不考虑其位数影响。
4. 第四大类 *Random()*为无需传参的函数，但考虑其功能特性（返回一个不小于 5000 的随机 unsigned int 型整数），及在程序中参与运算的方式（主要出现在等式右值处或函数调用时的形参处，提供其返回值参与运算），因此将其与常量一同介绍。

表 2.1 基本库 Basic 中的常量表

类型	名字	含义	位数
mpz_t	*p*	生成元 p	2^{1024}
	Elgamal_g	ElGamal 加密的 g	
	Pedersen_g	Pedersen 承诺的 g	
	Pedersen_h	Pedersen 承诺的 h	
unsigned int	*A_Class*	L'优'	2^{32}
	B_Class	L'良'	
	C_Class	L'中'	
	D_Class	L'差'	
gmp_randstate_t	*seed*	随机大整数生成器第二个参数	-
unsigned int	*Random()*	返回一个不小于 5000 的随机整数	2^{32}

2.2 结构体

考虑到 mpz_t 类型大整数在 C++程序中使用时：需要使用 *mpz_init()* 函数初始化，需要使用 *mpz_clear()*函数进行释放，并且使用 ElGamal 公开密钥密码体制加密发送的密文为两个 mpz_t 大整数（在后文统一记为*(C1, C2)*的形式，以此形式表示一个 mpz_t 大整数对），为便于使用和传递，在基本库中使用 C++类定义 Message，以虚拟实现一个无需主动 *mpz_init()*初始化和 *mpz_clear()*释放的结构体。其成员将相关信息如下表 2.2 所示。

表 2.2 基本库 Basic 中的结构体 Message

含义	ElGamal 加密后的可解密文*(C1 , C2)*	
功能	实为 class，构造函数自动进行初始 *mpz_inits(C1,C2)*，便于使用和传递	
位数	$(2^{1024}, 2^{1024})$	
成员		
变量	类型	位数
C1	mpz_t	2^{1024}
C2	mpz_t	2^{1024}

2.3 基本定理

ElGamal 公钥密码体制、Pedersen 承诺相关方法以定理形式给出，且均满足安全性需求不再另行证明。在以下的 4 条定理的计算公式中，每进行一次乘、幂运算都会对 p 取模，为简化表达式取模运算未写出，仅在每公式最后用*(mod p)*进行表示。

定理 1. 公钥计算公式。对于给定密钥key，公钥：

$$public\_key = Elgamal\_g^{key}(mod\ p)$$

定理 2. ElGamal 加密计算公式。对于给定公钥 *public_key*、明文 *M*：

$$C_1 = g^r(mod\ p)$$
$$C_2 = M * g^r(mod\ p)$$

其中 *r* 为随机数，密文即为(C_1, C_2)。

定理 3. ElGamal 解密计算公式。对于给定私钥 *key*，(C_1, C_2)为用其对应公钥加密的密文，明文：

$$M = C_2 * C_1^{-x}(mod\ p)$$

定理 4. Pedersen 承诺计算公式。对于给定明文 *M*，Pedersen 承诺：

$$C = Pedersen\_g^r * Pedersen\_h^M(mod\ p)$$

其中 *r* 为随机数。

2.4 基本函数

Basic 提供六组基本函数（每组函数包含 1 至 4 个同构函数），分别用作进行“初始化密钥对”、“向文件发送信息”、“从文件接收信息”、“ElGamal 加密”、“ElGamal 解密”、“计算 Pedersen 承诺”，均需使用 C++GMP 库中部分函数。具体信息如下：

1、初始化密钥对

该组函数名为$Creat_Key()$。

首先通过 *Random()*随机产生私钥的值，并通过引用访问保存至 *key*，然后利用**定理 1** 计算出公钥的值，同样通过引用访问保存值至 *public_key*。改组共 1 个函数，其参数及说明如下表 2.3 所示。

表 2.3 *Creat_Key()***参数表**

序号	参数 1 含义 类型	参数 2 含义 类型
1	私钥保存位置 unsigned int&	公钥保存位置 mpz_t&

2、向文件发送信息

该组函数名为$Send_Message()$。

向传入的绝对路径文件（为信息接收方的文件夹下）中写入要发送的信息，从而实现信息的传出。该组根据要发送的信息的数据类型及数据个数的不同共有 4 个同构函数，其参数及说明如下表 2.4 所示。函数序号 1、2 的第二项参数中的“信息”在程序中均为加密后的密文，保障假设信息被窃听后窃听方无法得到和 Alice、Bob 成绩相关的信息；而第二项参数中的“成绩”、“私钥”为按照任务书及教师说明直接发送的未加密明文（是为提高课程设计可执行度，做出的这两项信息在游戏开始前初始化阶段能安全发送的假设）。

表 2.4 *Send_Message()***参数表**

序号	参数 1 含义 类型	参数 2 含义 类型	参数 3 含义 类型
1	"对方文件夹//文件名.txt" string	信息 mpz_t&	-
2	"对方文件夹//文件名.txt" string	信息 Message&	
3	"对方文件夹//文件名.txt" string	成绩 unsigned int	
4	"对方文件夹//文件名.txt" string	私钥 unsigned int	公钥 mpz_t&

3、从文件接收信息

该组函数名为$Recieve_Message()$。

从传入的相对路径文件（为我方信息接收方文件夹下）中读取要接收的信息，从而实现信息的接收。该组根据要接收的信息的数据类型及数据个数的不同共有 4 个同构函数，其参数及说明如下表 2.5 所示。所有参数类型都为引用，实现函数通过引用访问直接将接收到的内容保存至原参数。

表 2.5 *Recieve_Message()***参数表**

序号	参数 1 含义 类型	参数 2 含义 类型	参数 3 含义 类型
1	"文件名.txt" const char*	信息保存位置 mpz_t&	-
2	"文件名.txt" const char*	信息保存位置 Message&	
3	"文件名.txt" const char*	成绩保存位置 unsigned int&	
4	"文件名.txt" const char*	私钥保存位置 unsigned int&	公钥保存位置 mpz_t&

4、ElGamal 加密

该组函数名为$ElGamal_Lock()$。

使用传入的随机数 *random*、公钥 *public_key*，将传入的明文 *send* 利用**定理 2** 进行 ElGamal 加密，并将得到的密文保存至“密文保存位置”变量 *Message*（该变量数据类型为 mpz_t 或 Message，函数通过其引用进行访问）。该组根据要加密的信息的数据类型、及要得到的密文数据类型不同，共有 4 个同构函数，其参数及说明如下表 2.6 所示。

其中，当第四项参数的类型为 mpz_t&时，函数只计算**定理 2** 中的$C_2 = M * g^r (mod\ p)$，并将该值保存至作为密文保存位置的变量中，此时所得的 ElGamal 密文时不完整的，只能起到对明文进行加密的作用，无法使用**定理 3** 进行 ElGamal 解密。

表 2.6 *ElGamal_Lock()***参数表**

序号	参数 1 含义 类型	参数 2 含义 类型	参数 3 含义 类型	参数 4 含义 类型
1	随机数 unsigned int	公钥 mpz_t&	明文 unsigned int	密文保存位置 Message&
2	随机数 unsigned int	公钥 mpz_t&	明文 unsigned int	密文保存位置 mpz_t&
3	随机数 unsigned int	公钥 mpz_t&	明文 mpz_t&	密文保存位置 Message&
4	随机数 unsigned int	公钥 mpz_t&	明文 mpz_t&	密文保存位置 mpz_t&

5、ElGamal 解密

该组函数名为$ElGamal_Unlock()$。

使用传入的私钥 *key*，将传入的密文 *Message* 利用**定理 3** 进行解密，并将解得的明文值保存值“明文保存位置”（数据类型为 mpz_t，函数通过其引用进行访问）。该组共有 4 个函数，其参数及说明如下表 2.7 所示。

表 2.7 *ElGamal_Unlock()***参数表**

序号	参数 1 含义 类型	参数 2 含义 类型	参数 3 含义 类型
1	私钥 unsigned int	密文 Message&	明文保存位置 mpz_t&

6、计算 Pedersen 承诺

该组函数名为$Pedersen_Commit()$。

使用传入的随机数 *random*，将传入的明文 *send* 利用**定理 4** 计算 Pedersen 承诺，并将计算得的承诺值保存至“承诺保存位置”（数据类型为 mpz_t，函数通过其引用进行访问）。该组根据要加密的信息的数据类型、及使用的随机数数据类型不同，共有 2 个同构函数，其参数及说明如下表 2.8 所示。

表 2.8 *Pedersen_Commit ()***参数表**

序号	参数 1 含义 类型	参数 2 含义 类型	参数 3 含义 类型
1	随机数 unsigned int	明文 unsigned int	承诺保存位置 mpz_t&
2	随机数 mpz_t&	明文 mpz_t&	承诺保存位置 mpz_t&

3 “约定参数”的概念及实现

3.1 定义

在情景二及情景三中，本组设计方案中使用到“只有 Alice 和 Bob 知道的参数”、“只有 Alice 和第三方知道的参数”、“只有 Alice、Bob 和第三方知道的参数”等概念限定的参数。提出以下定义：

定义 1. 约定参数 *r*。①甲方和乙方的约定参数 *r*，是指游戏中只有甲方和乙方知道其真实值的一个 mpz_t 类型随机大整数；②该约定参数 *r* 为约定方的私有参数；③约定参数 *r* 在游戏开始前的初始化阶段实现；④首先由约定方中一方通过随机数函数产生一个随机数 *r*，然后使用对方公钥（该密钥对是仅在约定参数时使用的临时局部变量）进行 ElGamal 加密，将密文发送给对方后，对方用其私钥进行 ElGamal 解密，从而完成了一个约定参数 *r* 的产生；⑤产生约定参数 *r* 的一方称为“发送方”，接收约定参数 *r* 的一方称为“接收方”。

说明：

1. 安全性：在上述**定义 1** 中，由于约定参数 *r* 产生自随机函数，且产生方将 *r* 的值发送给其余约定方时，使用 ElGamal 公钥密码体制进行信息加密，从而确保了游戏中只有约定方得到该约定参数 *r* 的真实值。即使存在监听方，也无法从使用 ElGamal 密钥加密得到的密文中推算出明文（约定参数 *r*）的真实值。因此，约定参数概念的产生及实现均满足安全性需求。

2. 在程序中，用约定参数 *r* 的下标区分约定方，A 表示 Alice，B 表示 Bob，C 表示第三方。例如 r_{AB} 表示 Alice 和 Bob 的约定参数，r_{AC} 表示 Alice 和 Bob 的约定参数。若为第三方、Alice 和 Bob 三方的约定参数，则用 r_3 表示。

3.2 实现

例 1.1. 约定参数 r_{AB}的具体实现：

1. Bob 产生一对自己的密钥，把公钥通过文件发送给 Alice；
2. Alice 产生一个随机大整数r_{AB}；
3. Alice 用 Bob 的公钥加密r_{AB}得到密文，将密文通过文件发送给 Bob；
4. Bob 收到文件后，用自己的密钥解密，即可得到r_{AB}。

例 1.2. 约定参数 r_3的具体实现：

1. Alice 产生一对自己的密钥,把公钥通过文件发送给第三方；
2. Bob 产生一对自己的密钥，把公钥通过文件发送给第三方；
3. 第三方产生一个随机大整数r_3；
4. 第三方用 Alice 的公钥加密r_3，将得到的密文发送给 Alice；用 Bob 的公钥加密r_3，得到密文发送给 Bob；
5. Alice 接收文件后用自己的密钥解密得到r_3；
6. Bob 收到文件后，用自己的密钥解密得到r_3。

该过程在程序的函数实现中，初始化阶段，发送方调用：

$bool\ Basic::Send\_Agreement\_Number()$

接收方在初始化阶段则需调用：

$bool\ Basic::Recieve\_Agreement\_Number()$

两函数的伪代码如下：

算法 1.1. *Send_Agreement_Number*

输入：string *my_name*, string *people_name*, mpz_t& *Number*
含义分别为发送方名称（如 Alice、Bob、Third）、接收方名称、要发送的约定参数。

输出：安全地完成向接收方发送约定参数密文

```
file_name = "SAN_" + people_name + "_Temp_Public_Key.txt";
    WHILE(Recive_Message(file_name, t1) == FALSE)
     SLEEP(1000);
file_name = people_name + "\\\\SAN_" + my_name + "_Temp_Send.txt";
Message send;
ElGamal_Lock(Random(), t1, Number, send);
IF (Send_Message(file_name, send) == FALSE)
    RETURN FALSE;
RETURN TRUE;
```

说明：

1. 先接收 *people_name* 的临时公钥,放于 *t1*
2. 对方公钥统一命名为：
   "SAN_Xx_Temp_Public_Key.txt"
3. 将约定数用 *people_name* 公钥按**定理 2** 加密
4. 发送的文件统一路径及命名为：
   "*people_name*\\SAN_Xx_Temp_Send.txt"

算法 1.2. *Recive_Agreement_Number*

输入：string *my_name*, string *people_name*, mpz_t& *Number*
含义分别为接收方名称（如 Alice、Bob、Third）、发送方名称、约定参数保存位置。

输出：安全地完成从发送方接收约定参数

```
temp_key = Random();
Creat_Key(temp_key, t1);
file_name = people_name + "\\\\SAN_" + my_name + "_Temp_Public_Key.txt";
IF (Send_Message(file_name, t1) ==FALSE)
 RETURN FALSE;
Message receive;
file_name = "SAN_" + people_name + "_Temp_Send.txt";
WHILE (Recive_Message(file_name, receive) == FALSE)
 SLEEP(1000);
ElGamal_Unlock(temp_key, receive, Number);
RETURN TRUE;
```

说明：

1. 先产生临时随机私钥 *temp_key*，按**定理 1** 计算出公钥放于 *t1*
2. 发送公钥 *t1* 给 *people_name*，文件统一命名为：
   "*people_name*\\SAN_Xx_Temp_Public_Key.txt"
3. 接收密文，文件统一命名为：
   " SAN_Xxx_Temp_Send.txt"
4. 用私钥 *temp_key* 按**定理 3** 解密得到约定参数

4 各场景重叠数据汇总

4.1 公有参数表

各场景使用相同的参数与函数基本库 Basic，因此，三个场景的参数表中均包含部分上文中的表 2.1（基本库 Basic 中的常量表）中内容。另外不考虑表 2.1 中 *seed* 的位数影响，因此在后文的所有分析中将不对其进行考虑。

各场景使用情况：

在不考虑初始化阶段的情况下，其中：

场景 1 使用到 *p*、*Elgamal_g*、*Random()*；

场景 2 使用到 *p*、*Elgamal_g*、*Random()*;

场景 3 使用到 *p*、*Pedersen_g*、*Random()*。

4.2 公有函数伪代码

1、基本库 Basic 中函数

各场景使用相同的参数与函数基本库 Basic，因此，三个场景使用到的函数均包含上文 2.4 基本函数一节中表 2.3 至表 2.8 中部分函数。

安全性：该节函数均较为简单，是对初始化密钥对、向文件发送信息、从文件接收信息、ElGamal 加密、ElGamal 解密、计算 Pedersen 承诺等基本操作的程序实现，因此，此部分函数均满足安全性需求，不在另作证明。

每组函数的各同构体对于参数的不同在具体代码编写时需采用相应的函数进行操作，在此仅展示该组函数的操作流程，不将具体实现时的函数区别作体现。此 6 组函数伪代码如下，输入输出含义及相关说明参见上文 2.4 基本函数一节。

算法 2.1. *Creat_Key*

输入：unsigned int& *key*, mpz_t& *public_key*

输出：初始化一对密钥，并保存至 *key*、*public_key*

```
key = Random();
mpz_powm_ui(public_key, Elgamal_g, key, p);
```

算法 2.2. *Send_Message*

输入：string *file_name, Message*

输出：将 *Message* 输出至 *file_name* 路径文件中

```
FILE* outfile = fopen(file_name, "w");
IF(!outfile) RETURN FALSE;
mpz_out_str(outfile, 37, Message);
fclose(outfile);
RETURN TRUE;
```

说明：仅在文件输出成功时返回 TRUE

算法 2.3. *Recive_Message*

输入：const char* *file_name*, *Message*

输出：从 *file_name* 路径文件中读入，并保存至 *Message*

```
fstream infile(file_name, ios::in);
IF (!infile) RETURN FALSE;
mpz_set_str(Message, temp, 37);
infile.close();
remove(file_name);
RETURN TRUE;
```

说明：仅在文件读取成功时返回 TRUE

算法 2.4. *ElGamal_Lock*

输入：*unsigned int random, mpz_t& public_key, send, Message*

输出：将明文 ElGamal 加密得到的密文保存至 *Message*

```
mpz_powm_ui(Message.C1, Elgamal_g, random, p);
mpz_powm_ui(t1, public_key, random, p);
mpz_mul(t2, t1, send);
mpz_mod(Message.C2, t2, p);
```

算法 2.5. *ElGamal_Unlock*

输入：unsigned int *key*, Message& *Message*, mpz_t& *received*

输出：将密文 ElGamal 揭秘得到的明文保存至 *received*

```
mpz_powm_ui(t2, Message.C1, key, p);
mpz_invert(t3, t2, p);
mpz_mul(t1, Message.C2, t3);
mpz_mod(recived, t1, p);
```

算法 2.6. *Pedersen_Commit*

输入：*random*, *send*, mpz_t& *Message*

输出：计算得到 Pedersen 承诺并保存至 *Message*

```
mpz_powm(t1, Pedersen_g, random, p);
mpz_powm(t2, Pedersen_h, send, p);
mpz_mul(t3, t1, t2);
mpz_mod(Message, t3, p);
```

各场景使用情况：

在不考虑初始化阶段的情况下，其中：

场景 1 使用到 *Send_Message*、*Recive_Message*、*ElGamal_Lock*、*ElGamal_Unlock*；

场景 2 使用到 *Send_Message*、*Recive_Message*、*ElGamal_Lock*、*ElGamal_Unlock*；

场景 3 使用到 *Send_Message*、*Recive_Message*、*ElGamal_Lock*。

2、主控类 People 中函数

People 类，是为对游戏中各端进程进行控制，而采用的函数及变量集合的封装。各场景游戏中，各端 People 中使用的*bool Menu*()菜单（成绩选择）

函数、*bool Choose*()选择是否作弊函数，在此给出其伪代码。

算法 3.1. *Menu*

输入：*c*

从键盘的用户输入获取用户选择。

输出：*true / false*

初始化私有参数：*grade*，并依据 *c* 的不同，返回不同的 bool 值。

```
cout << 显示输入指导界面;
char c; cin >> c;
 WHILE (c < '0' || c > '4')
 {
      cout << "输入错误，请重新选择";
      cin.ignore(1024, '\n');
      cin >> c;
 }
 cin.ignore(1024, '\n');
 IF (c == '0') RETURN FALSE;
 ELSE IF (c == '1') grade = basic.A_Class;
 ELSE IF (c == '2') grade = basic.B_Class;
 ELSE IF (c == '3') grade = basic. C_Class;
 ELSE grade = basic.D_Class;
 RETURN TRUE;
```

说明：

1. 将键盘的输入的存储变量类型设置为 char，且在每次输入后都使用 *ignore()*将缓冲区清空，便可做到选择菜单能容许用户的任何种类的错误输入且进行纠正，保障程序不会因为输入问题发生崩溃。
2. 程序中的输入指导界面中，提示用户不同选项所对应的不同按键输入，对应关系分别为：
   1 对应“成绩选择 优”
   2 对应“成绩选择 良”
   3 对应“成绩选择 中”
   4 对应“成绩选择 差”
   0 对应“退出游戏”
3. *Menu()*会根据 *c* 的值初始化 *grade* 值。*Menu()*和 *grade* 均为 People 类成员，因此 *Menu()*可对 *grade* 进行直接访问，无需显式地进行参数传递。
4. 该算法显示的是 Alice 和 Bob 端的菜单情况，有成绩选择选项，第三方端无成绩选择选项，0 对应“退出游戏”，1 对应“开始游戏”，其流程与本算法类似，不再作为单独算法列出。其伪代码与下文的算法 3.2 相同，仅在具体实现时采用不同输入指导界面。

算法 3.2. *Choose*

输入：*c*

从键盘的用户输入获取用户选择。

输出：*true / false*

依据 *c* 的不同，返回不同的 bool 值。

```
cout << 显示输入指导界面;
char c; cin >> c;
 WHILE (c < '0' || c > '1')
 {
      cout << "输入错误，请重新选择";
      cin.ignore(1024, '\n');
      cin >> c;
 }
 cin.ignore(1024, '\n');
 IF (c == '0') RETURN FALSE;
 RETURN TRUE;
```

说明：

1. 本函数是在把“游戏公平性”加入考虑范围的情况下，使 Alice 或 Bob 端在游戏中需要发送信息时，能够选择发送真实信息、还是使用随机数生成函数随机生成数字进行发送，而加入的一个选择菜单函数。
2. 程序中的输入指导界面中，提示用户不同选项所对应的不同按键输入，对应关系分别为：
   1 对应“发送实际值”
   0 对应“CHEET”
3. 在无需考虑 Alice 和 Bob 会作弊时无需调用该函数判断，直接运行发送真实信息的代码即可，后文的场景一、场景二、场景三为考虑了 Alice 和 Bob 会作弊情况下的展示与分析。
4. 将键盘的输入的存储变量类型设置为 char，且在每次输入后都使用 *ignore()*将缓冲区清空，便可做到选择菜单能容许用户的任何种类的错误输入且进行纠正，保障程序不会因为输入问题发生崩溃。
5. 本次课程设计中不考虑第三方作弊（使用随机数代替真实信息的发送）的情况，因此第三方程序端的 People 主控类中不设置此函数。

5 场景一

在后文的三个场景分析中：
第三方程序端简称为 TTP，Alice 程序端简称为 Alice，Bob 程序端简称为 Bob。
各个场景使用到的 Basic 库参数，如上文 4.1 节公有参数表所说明，在各方参数表中不再重复列出。
各个场景使用到的 Basic 库函数的伪代码及安全性，如上文 4.2 节公有函数伪代码所说明，在各场景伪代码及安全性证明中不再重复列出。

5.1 参数表

场景一的游戏解决方案中，TTP、Alice、Bob 参数表分别如下表 5.1、表 5.2、表 5.3 所示，使用到的 Basic 库参数参见 4.1 节公有参数表。

表 5.1 场景一 TTP 参数表

名字	含义	位数
alice_grade	Alice 的成绩	2^{32}
bob_grade	Bob 的成绩	
alice_key	Alice 的私钥	
bob_key	Bob 的私钥	

表 5.2 场景一 Alice 参数表

名字	含义	位数
grade	Alice 的成绩	2^{32}
key	Alice 的私钥	
public_key	Alice 的公钥	2^{1024}

表 5.3 场景一 Bob 参数表

名字	含义	位数
grade	Bob 的成绩	2^{32}
key	Bob 的私钥	
public_key	Bob 的公钥	2^{1024}

5.2 程序交互过程图

场景一程序交互过程如下图 5 所示。图中“(n)”形式的标号为程序中步骤执行顺序，各步骤含义如下：
(1)$[S_A]_{pk_A} = (S_A, pk_A)$：Alice 使用自己的公钥加密自己的成绩，所得密文为$[S_A]_{pk_A}$
(1)$[S_B]_{pk_B} = (S_B, pk_B)$ ：Bob 使用自己的公钥加密自己的成绩，所得密文为$[S_B]_{pk_B}$
(2)Alice 和 Bob 将密文发送给 TTP
(3)TTP 使用sk_A解密 Alice 发送的密文，使用sk_B解密 Bob 发送的密文，判断解得密文是否与他们各自成绩相同。
(4)相同则二人没有作弊，将二人消息转发给各自的对方；不相同则有人作弊，可以发送终止游戏的命令提前终止游戏。
(5)没有人作弊时，Alice 和 Bob 使用自己的私钥解密 TTP 转发的对方密文，若解得结果与自己成绩相同，则对方成绩与自己相同；若解得结果与自己成绩不同则对方成绩与自己不同。

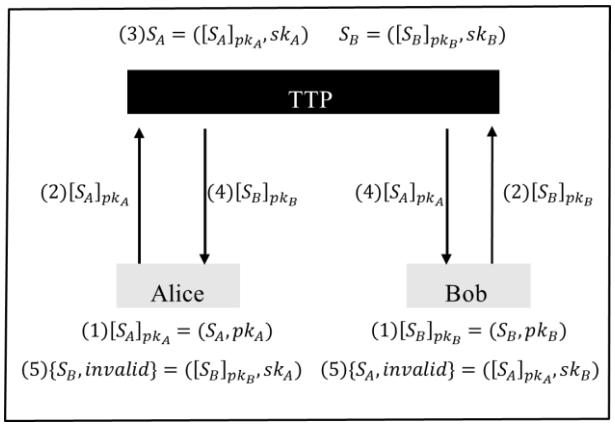

图 5 场景一程序交互过程图

5.3 伪代码

程序主要分为初始化阶段、游戏阶段 2 个阶段。

1、初始化阶段：

初始化阶段完成表 5.1、表 5.2、表 5.3 中的 TTP、Alice、Bob 参数表参数的初始化。

其中密钥对的初始化方法：TTP 根据 Alice、Bob 成绩是否相同，给两人分配相同或不同的密钥对，成绩相同分配相同密钥对，成绩不同分配不同密钥对。

各端伪代码如下文算法所示，TTP 的初始化阶段函数伪代码为算法 5.1.1，Alice 的初始化阶段函数伪代码为算法 5.1.2，Bob 的初始化阶段函数伪代码为算法 5.1.3。

算法 5.1.1. TTP-Init

输入：无

输出：初始化 TTP 的参数表，并发送密钥对给 Alice、Bob

```
    WHILE (Recive_Message("Alice_Grade.txt", alice_grade)
== FALSE)
        Sleep(1000);
    WHILE (Recive_Message("Bob_Grade.txt", bob_grade)
```

```
== FALSE)
        Sleep(1000);
    Creat_Key(alice_key, alice_public_key);
    IF (alice_grade != bob_grade)
        Creat_Key(bob_key, bob_public_key);
    ELSE {Bob 的密钥对 = Alice 的密钥}
Send_Message("Alice//Key.txt", alice_key, alice_public_key);
Send_Message("Bob//Key.txt", bob_key, bob_public_key);
```

说明：

1. TTP 的初始化首先接收 Alice、Bob 的成绩
2. 然后产生使用 *Creat_Key()*函数产生一对密钥并将其发送给 Alice
3. 然后比较 Alice 和 Bob 成绩是否相同。
   a) 相同：Bob 密钥对与 Alice 密钥对相同
   b) 不同：再使用 *Creat_Key()*函数产生一对不同的密钥对。
4. 将 Bob 密钥对发送给 Bob。

算法 5.1.2.　Alice-Init

输入：已经初始化好的S_A

输出：发送成绩给 TTP，并初始化 Alice 的参数表

```
    Send_Message("Third\\Alice_Grade.txt", grade);
    WHILE (Recive_Message("Key.txt", key, public_key) ==
FALSE)
    Sleep(1000);
```

说明：

1. Alice 的初始化首先将自己成绩发送给 TTP
2. 然后等待接收 TTP 发送的我方密钥对

算法 5.1.3.　Bob-Init

输入：已经初始化好的S_B

输出：发送成绩给 TTP，并初始化 Bob 的参数表

```
    Send_Message("Third\\Bob_Grade.txt", grade);
    WHILE (Recive_Message("Key.txt", key, public_key) ==
FALSE)
    Sleep(1000);
```

1. Bob 的初始化首先将自己成绩发送给 TTP
2. 然后等待接收 TTP 发送的我方密钥对

2、游戏阶段：

游戏阶段用程序实现图 5 中的交互过程。

各端伪代码如下文算法所示，TTP 的游戏阶段函数伪代码为算法 5.2.1，Alice 的游戏阶段函数伪代码为算法 5.2.2，Bob 的游戏阶段函数伪代码为算法 5.2.3。

算法 5.2.1.　TTP-Game

```
输入：无
输出：完成场景一的游戏过程
WHILE (1)
{   IF (Menu() == FALSE) BREAK;
    Init();
    WHILE (Recive_Message("Alice_Send.txt", recive_alice)
== FALSE)
        Sleep(1000);
    ElGamal_Unlock(alice_key, recive_alice, recive);
     IF (recive != alice_grade) {终止游戏; RETURN; }
    WHILE (Recive_Message("Bob_Send.txt", recive_bob) ==
FLASE)
        Sleep(1000);
    ElGamal_Unlock(bob_key, recive_bob, recive);
    IF (recive != bob_grade) {终止游戏; CONTINUE; }
    ELSE
    {
        Send_Message("Alice\\Third_Send.txt", recive_bob);
        Send_Message("Bob\\Third_Send.txt", recive_alice);
    }
}
```

说明：

1. 最外层的 while (1)配合 *Menu()*菜单，实现 *Menu()*返回 false 即选择“退出游戏”时终止程序（即跳出该循环）的目的。
2. TTP 的一次游戏中，首先 *Menu()*选择是否退出游戏。
3. 然后调用 *Init()*函数完成初始化阶段
4. 然后等待接收 Alice、Bob 发送的密文，接收到后用两人对应私钥解密，判断解得密文是否与他们各自成绩相同。
5. 相同则二人没有作弊，将二人消息转发给各自的对方；不相同则有人作弊，发送终止游戏的命令。

算法 5.2.2.　Alice-Game

```
输入：无
输出：完成场景一的游戏过程
WHILE (1)
{   IF (Menu() == FALSE) BREAK;
    Init();
```

```
    IF (Choose() == TRUE)
        ElGamal_Lock(Random(), public_key, grade, send2);
    ELSE
    {
        mpz_urandomm(send2.C1, seed, p);
        mpz_urandomm(send2.C2, seed, p);
    }
    Send_Message("Third\\Alice_Send.txt", send2);
    WHILE (1)
    {
        IF ((remove("Cheet_Warning.txt") != EOF))
            { 收到 TTP 的游戏终止命名; CONTINUE; }
        IF (Recive_Message("Third_Send.txt", recive2) ==
TRUE)
        {
            ElGamal_Unlock(key, recive2, recive);
            IF (recive == grade){成绩相同界面; }
            ELSE {显示成绩不同界面; }
        }
    }
}
```

说明：

1. 最外层的 while (1)配合 *Menu()*菜单，实现 *Menu()*返回 false 即选择“退出游戏”时终止程序（即跳出该循环）的目的。
2. Alice 的一次游戏中，首先 *Menu()*选择其成绩或是选择退出游戏。
3. 然后调用 *Init()*函数完成初始化阶段
4. 然后 *Choose()*选择发送真实信息、还是使用随机数生成函数随机生成数字进行发送，选择后发送相应信息。
5. 然后等待 TTP 发送的信息。若接收到 "Cheet_Warning.txt"，则有人作弊，提前终止游戏。
6. 若接收到 TTP 转发的 Bob 的密文，则无人作弊，用自己的私钥进行解密，将解得的信息与自己的成绩进行比较。
7. 相同则显示二人成绩相同的界面，不同则显示二人成绩不同的界面。

算法 5.2.3. Bob-Game

输入：无

输出：完成场景一的游戏过程

```
WHILE (1)
{   IF (Menu() == FALSE) BREAK;
    Init();
    IF (Choose() == TRUE)
        ElGamal_Lock(Random(), public_key, grade, send2);
    ELSE
    {
        mpz_urandomm(send2.C1, seed, p);
        mpz_urandomm(send2.C2, seed, p);
    }
    Send_Message("Third\\Bob_Send.txt", send2);
    WHILE (1)
    {
        IF ((remove("Cheet_Warning.txt") != EOF))
            { 收到 TTP 的游戏终止命名; CONTINUE; }
        IF (Recive_Message("Third_Send.txt", recive2) ==
TRUE)
        {
            ElGamal_Unlock(key, recive2, recive);
            IF (recive == grade){成绩相同界面; }
            ELSE {显示成绩不同界面; }
        }
    }
}
```

说明：

1. 最外层的 while (1)配合 *Menu()*菜单，实现 *Menu()*返回 false 即选择“退出游戏”时终止程序（即跳出该循环）的目的。
2. Bob 的一次游戏中，首先 *Menu()*选择其成绩或是选择退出游戏。
3. 然后调用 *Init()*函数完成初始化阶段
4. 然后 *Choose()*选择发送真实信息、还是使用随机数生成函数随机生成数字进行发送，选择后发送相应信息。
5. 然后等待 TTP 发送的信息。若接收到 "Cheet_Warning.txt"，则有人作弊，提前终止游戏。
6. 若接收到 TTP 转发的 Alice 的密文，则无人作弊，用自己的私钥进行解密，将解得的信息与自己的成绩进行比较。
7. 相同则显示二人成绩相同的界面，不同则显示二人成绩不同的界面。

5.4 安全性证明

性质 1：假设 Alice 和 Bob 不会篡改自己的成绩，那么他们在游戏中一定能得到正确的判断结果，并且无法知道对方成绩。

证明.

在双方没有作弊的情况下，TTP 会将双方发送的信息转发给对方，因此，对于 Alice 和 Bob 端接收到的信息为：

Alice：$[S_B]_{pk_B}$

Bob 使用其公钥加密的 Bob 成绩。

Bob：$[S_A]_{pk_A}$

Alice 使用其公钥加密的 Alice 成绩。

根据 TTP 在初始化阶段密钥对的分配方法：TTP 根据 Alice、Bob 成绩是否相同，给两人分配相同或不同的密钥对，成绩相同分配相同密钥对，成绩不同分配不同密钥对。

因此有以下两种情况：

Alice、Bob 成绩相同时：Alice 和 Bob 密钥对为同一对，因此可使用自己的私钥解开对方发送的密文，即可得到明文。而假设中，Alice、Bob 不会篡改自己的成绩，因此，此时明文必为自己的成绩。此时得到正确的判断结果。

Alice、Bob 成绩不同时：Alice 和 Bob 密钥对不为同一对，因此使用自己的私钥解对方发送的密文后无法得到对方的明文，得到结果与自己成绩不同。此时得到正确的判断结果。并且，定理 2 保障了此时无法破解对方密文，即满足无法知道对方成绩。

综上所述：假设 Alice 和 Bob 不会篡改自己的成绩，那么他们在游戏中一定能得到正确的判断结果，并且无法知道对方成绩。

证毕.

性质 2：除了 Alice、Bob、第三方之外的任何人不会在游戏中得到与成绩相关的任何信息。

证明.

若存在监听方，则其可获得所有游戏中以文件形式发送的信息。则在游戏中其可获得的信息有：

$$[S_A]_{pk_A}、[S_B]_{pk_B}$$

因其缺失 Alice 公钥所对应的私钥、Bob 公钥所对应的私钥，因此定理 2 保障了此时无法破解密文，即无法获取任何有关成绩的信息。

证毕.

5.5 性能分析

在程序运行所在的 64 位计算机中，基本存储单位：

$$1Bit = 64bit$$

因此，在后续的性能分析中，将使用：2^{32} 位数对应 4 比特，2^{1024} 位数对应 128 比特。

对于假定 k 比特的整数 p，各操作时间复杂度如下：

$m^c\ (mod\ p)$时间复杂度：$O(\ (\log c) * k^2\ )$

$(m_1 * m_2)(mod\ p)$时间复杂度：$O(k^2)$

$m^{-1}\ (mod\ p)$时间复杂度：$O(k^3)$

产生随机数时间复杂度：$O(1)$

时间复杂度：

各端游戏中使用到的操作如下表 T1 所示。其中模幂运算中的指数在场景一中为 2^{32} 位数。

表 T1 场景一操作次数统计

	模幂	乘模	乘逆	随机数
Alice	3	2	1	1
Bob	3	2	1	1
TTP	2	2	2	0

因此程序时间复杂度为：

$$T(n) = 8 \times O(\ (\log 4) * k^2\ ) + 6 \times O(k^2) + 4 \times O(k^3) +$$
$$= 4 \cdot O(k^3) + (8log4 + 6) \cdot O(k^2) + 2$$

空间复杂度：

各端参数表中有 2^{32} 位数 8 个，2^{1024} 位数 2 个

公共参数中 2^{32} 位数 2 个，2^{1024} 位数 6 个

使用中间变量 2^{1024} 位数 10 个

因此，变量所占空间复杂度：

$$S(n) = 18 \times 128Bit + 10 \times 4Bit = 2344Bit$$

6 场景二

6.1 参数表

场景二的游戏解决方案中，TTP、Alice、Bob 参数表分别如下表 6.1、表 6.2、表 6.3 所示，使用到的 Basic 库参数参见 4.1 节公有参数表。

表 6.1 场景二 TTP 参数表

名字	含义	位数
alice_grade	Alice 的成绩	2^{32}
alice_key	Alice 成绩对应私钥	
r3	三方约定参数	2^{1024}

表 6.2 场景二 Alice 参数表

名字	含义	位数
grade	Alice 的成绩	2^{32}
rAB	Alice、Bob 约定随机数	2^{1024}
r3	三方约定随机数	
commit	rAB、r3 算得的承诺	

表 6.3 场景二 Bob 参数表

名字	含义	位数
grade	Bob 的成绩	2^{32}
rAB	Alice、Bob 约定随机数	2^{1024}
r3	三方约定随机数	
commit	rAB、r3 算得的承诺	
public_key	Bob 成绩对应公钥	

6.2 程序交互过程图

场景二程序交互过程如下图 6 所示。图中“(n)”形式的标号为程序中步骤执行顺序，各步骤含义如下：

(1)$[g^{r_{AB}}h^{r_3}]\,pK_B$：Bob 使用自己的公钥加密承诺值，所得密文为$[g^{r_{AB}}h^{r_3}]\,pK_B$，然后将密文发送给 TTP。

(2)$M=(\,[g^{r_{AB}}h^{r_3}]\,pK_B,\,sK_A\,)$：TTP 使用 Alice 成绩对应的私钥解密得到的信息为M

(3)TTP 将 M 发送给 Alice 和 Bob

(4)Alice 和 Bob 分别各自将 TTP 发送的 M 与私有参数中的 $g^{r_{AB}}h^{r_3}$进行判断：相同则表示 Alice 成绩所对应私钥与 Bob 成绩所对应公钥为同一对，则二者成绩相同；否则 b 二者成绩不同。

没有人作弊时，Alice 和 Bob 均可得到正确结论；而只要出现作弊，二者均得到“对方成绩与自己成绩不同”的错误结论。

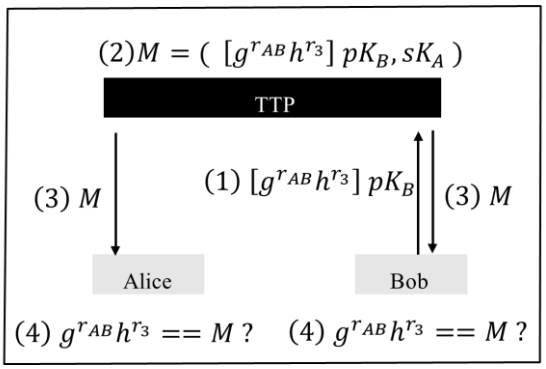

图 6 场景二程序交互过程图

6.3 伪代码

程序主要分为初始化阶段、游戏阶段 2 个阶段。

1、初始化阶段：

初始化阶段完成表 6.1、表 6.2、表 6.3 中的 TTP、Alice、Bob 参数表参数的初始化。

各端伪代码如下文算法所示，TTP 的初始化阶段函数伪代码为算法 6.1.1，Alice 的初始化阶段函数伪代码为算法 6.1.2，Bob 的初始化阶段函数伪代码为算法 6.1.3。

算法 6.1.1. TTP-Init

输入：无

输出：初始化 TTP 的参数表，并调用 *Creat_Key()*产生 4 对和成绩对应的不同密钥对，将所有公钥都发送给 Bob

```
WHILE (Recive_Message("Alice_Grade.txt", alice_grade) == FALSE)
        Sleep(1000);
mpz_urandomm(r3, seed, p);
Send_Agreement_Number("Third", "Alice", r3);
Send_Agreement_Number("Third", "Bob", r3);
Creat_Key(key, public_key);
IF (alice_grade == A_Class) alice_key = key;
Send_Message("Bob\\A_Class_Public_Key.txt", public_key);
Creat_Key(key, public_key);
IF (alice_grade == B_Class) alice_key = key;
Send_Message("Bob\\B_Class_Public_Key.txt", public_key);
Creat_Key(key, public_key);
IF (alice_grade == C_Class) alice_key = key;
Send_Message("Bob\\C_Class_Public_Key.txt",
```

public_key);

Creat_Key(key, public_key);

IF (*alice_grade* == *D_Class*) *alice_key* = *key*;

Send_Message("Bob\\D_Class_Public_Key.txt", *public_key*);

说明：

1. TTP 的初始化首先接收 Alice 成绩
2. 然后产生约定参数 *r3*，使用 *Send_Agreement_Number()*将约定参数 *r3* 发送给 Alice、Bob。
3. 然后 *Creat_Key()*函数产生 4 对和成绩对应的不同密钥对，并将所有公钥都发送给 Bob
4. 4 对密钥中，选取与 Alice 成绩所对应的那一对，用其私钥设置私有参数中的 alice_key。

算法 6.1.2.　Alice-Init

输入：已经初始化好的S_A

输出：发送成绩给 TTP，并初始化 Alice 的参数表

Send_Message("Third\\Alice_Grade.txt", *grade*);

mpz_urandomm(rAB, seed, p);

Send_Agreement_Number("Alice", "Bob", *rAB*);

Recieve_Agreement_Number("Alice", "Third", *r3*);

Pedersen_Commit(rAB, r3, commit);

说明：

1. Alice 的初始化首先将自己成绩发送给 TTP
2. 然后产生约定参数 *rAB*，使用 *Send_Agreement_Number()*将约定参数 *rAB* 发送给 Bob。
3. 然后使用 *Recieve_Agreement_Number()*从 TTP 接收约定参数 *r3*。
4. 最后使用 *rAB*、*r3* 计算 Pedersen 承诺保存在 *commit* 中。

算法 6.1.3.　Bob-Init

输入：已经初始化好的S_B

输出：初始化 Bob 的参数表

Recieve_Agreement_Number("Bob", "Alice", rAB);

Recieve_Agreement_Number("Bob", "Third", r3);

IF (*grade* == *A_Class*)

temp_file = "A_Class_Public_Key.txt";

ELSE IF (*grade* == *B_Class*)

temp_file = "B_Class_Public_Key.txt";

ELSE IF (*grade* == *C_Class*)

temp_file = "C_Class_Public_Key.txt";

ELSE *temp_file* = "D_Class_Public_Key.txt";

WHILE (*Recive_Message*(*temp_file*, *public_key*) == FALSE)

Sleep(1000);

Pedersen_Commit(rAB, r3, commit);

说明：

1. 首先使用 *Recieve_Agreement_Number()*从 Alice 获取约定参数 *rAB*。
2. 然后使用 *Recieve_Agreement_Number()*从 TTP 接收约定参数 *r3*。
3. 然后等待 TTP 发送 4 个成绩所对应的公钥，选择自己成绩所对应的那一个保存为自己的 *public_key*。
4. 最后使用 *rAB*、*r3* 计算 Pedersen 承诺保存在 *commit* 中。

2、游戏阶段：

游戏阶段用程序实现图 6 中的交互过程。

各端伪代码如下文算法所示，TTP 的游戏阶段函数伪代码为算法 6.2.1，Alice 的游戏阶段函数伪代码为算法 6.2.2，Bob 的游戏阶段函数伪代码为算法 6.2.3。

算法 6.2.1.　TTP-Game

输入：无

输出：完成场景二的游戏过程

WHILE (1)

{　IF (*Menu*() == FALSE) BREAK;

Init();

WHILE (*Recive_Message*("Bob_Send.txt", *recive2*) == FALSE)

Sleep(1000);

ElGamal_Unlock(alice_key, recive2, send);

Send_Message("Alice\\Third_Send.txt", *send*);

Send_Message("Bob\\Third_Send.txt", *send*);

}

说明：

1. 最外层的 while (1)配合 *Menu()*菜单，实现 *Menu()*返回 false 即选择“退出游戏”时终止程序（即跳出该循环）的目的。
2. TTP 的一次游戏中，首先 *Menu()*选择是否退出游戏。

3. 然后调用 *Init()*函数完成初始化阶段
4. 然后等待接收 Bob 发送的密文，接收到后用 Alice 成绩对应的私钥解密，将得到的信息保存至 *send*。
5. 最后将 *send* 发送给 Alice 和 Bob。

算法 6.2.2. Alice-Game

输入：无

输出：完成场景二的游戏过程

```
WHILE (1)
{   IF (Menu() == FALSE) BREAK;
    Init();
    WHILE (Recive_Message("Third_Send.txt", recive) == FALSE)
        Sleep(1000);
    IF (commit == recive) {成绩相同界面; }
    ELSE {显示成绩不同界面; }
}
```

说明：

1. 最外层的 while (1)配合 *Menu()*菜单，实现 *Menu()*返回 false 即选择“退出游戏”时终止程序（即跳出该循环）的目的。
2. Alice 的一次游戏中，首先 *Menu()*选择其成绩或是选择退出游戏。
3. 然后调用 *Init()*函数完成初始化阶段
4. 然后等待 TTP 发送的信息
5. 将收到信息与自己计算的 *commit* 进行比较，相同成绩相同，显示二人成绩相同的界面，不同则显示二人成绩不同的界面。

算法 6.2.3. Bob-Game

输入：无

输出：完成场景二的游戏过程

```
WHILE (1)
{   IF (Menu() == FALSE) BREAK;
    Init();
     IF (Choose() == TRUE)
      ElGamal_Lock(Random(), public_key, commit, message);
     ELSE
     {
          mpz_urandomm(message.C1, seed, p);
          mpz_urandomm(message.C2, seed, p);
     }
     Send_Message("Third\\Bob_Send.txt", message);
     WHILE (Recive_Message("Third_Send.txt", recive) == FALSE)
          Sleep(1000);
    IF (commit == recive) {成绩相同界面; }
    ELSE {显示成绩不同界面; }
}
```

说明：

1. 最外层的 while (1)配合 *Menu()*菜单，实现 *Menu()*返回 false 即选择“退出游戏”时终止程序（即跳出该循环）的目的。
2. Bob 的一次游戏中，首先 *Menu()*选择其成绩或是选择退出游戏。
3. 然后调用 *Init()*函数完成初始化阶段
4. 然后 *Choose()*选择发送真实信息、还是使用随机数生成函数随机生成数字进行发送，选择后发送相应信息。
5. 然后等待 TTP 发送的信息
6. 将收到信息与自己计算的 *commit* 进行比较，相同成绩相同，显示二人成绩相同的界面，不同则显示二人成绩不同的界面。

6.4 安全性证明

性质 1：假设 Alice 和 Bob 不会篡改自己的成绩，那么他们在游戏中一定能得到正确的判断结果，并且无法知道对方成绩。

证明.

在初始化阶段，Alice 和 Bob 都将各自计算出相同的承诺值：

$$commit = g^{r_{AB}} h^{r_3}$$

TTP 使用 Alice 成绩对应的私钥解密 Bob 发送的、使用 Bob 成绩对应的公钥加密的 *commit* 承诺值，将解得的此信息 *M* 发送给双方，因此对于 Alice 和 Bob 端接收到的信息为：

Alice：$M = (\ [g^{r_{AB}} h^{r_3}]\ pK_B, sK_A\ )$

Bob：$M = (\ [g^{r_{AB}} h^{r_3}]\ pK_B, sK_A\ )$

因此有以下两种情况：

Alice、Bob 成绩相同时：Alice 和 Bob 密钥对为同一对，因此可使用 Alice 成绩随对应的私钥解开 Bob 成绩所对应公钥加密的密文，即可得到正确明文 *commit*。此时得到$M == commit$。此时得到正确的判断结果。

Alice、Bob 成绩不同时：Alice 和 Bob 密钥对不为

同一对，因此使用 Alice 成绩所对应的私钥解 Bob 成绩所对应公钥加密的密文后无法得到对方的明文，此时得到$M! = commit$。此时得到正确的判断结果。

并且，Bob 没有获得任何 Alice 发送的信息，Alice 没有获得 Bob 的密文，且没有 Bob 成绩对应的私钥，即满足双方无法知道对方成绩。

综上所述：假设 Alice 和 Bob 不会篡改自己的成绩，那么他们在游戏中一定能得到正确的判断结果，并且无法知道对方成绩。

证毕.

性质 3：第三方不会在游戏中得到任何额外信息，并且除了 Alice、Bob、第三方之外的任何人不会在游戏中得到与成绩相关的任何信息。

证明.

对于第三方，其接收 Bob 发送的信息为：

$$[g^{r_{AB}}h^{r_3}]\,pK_B$$

因第三方缺失 *rAB*，由定理 4 及定义 1 知：第三方无法获取承诺 *commit* 的真实值，即使第三方使用 4 个私钥一一尝试解密，得到 4 个$M_i, i \in (1,4)$，第三方也无法对 4 个M_i任何区分。

因此第三方在游戏中无法获得任何和成绩有关的额外信息。

若存在监听方，则其可获得所有游戏中以文件形式发送的信息。则在游戏中其可获得的信息有：

$$[g^{r_{AB}}h^{r_3}]\,pK_B、M$$

因其缺失 Bob 公钥所对应的私钥，由定理 2 知，其无法通过$[g^{r_{AB}}h^{r_3}]\,pK_B$获取任何成绩有关信息。因其缺失 *rAB*、*r3*，由定理 4 及定义 1 知，监听方无法获取承诺 *commit* 的真实值，即无法通过 *M* 获取任何成绩有关信息。

综上所述：第三方不会在游戏中得到任何额外信息，并且除了 Alice、Bob、第三方之外的任何人不会在游戏中得到与成绩相关的任何信息。

证毕.

6.5 性能分析

时间复杂度：

各端游戏中使用到的操作如下表 T2 所示。其中模幂运算中的指数在场景二中为 2^{1024} 位数。

表 T2 场景二操作次数统计

	模幂	乘模	乘逆	随机数
Alice	0	0	0	0
Bob	2	1	0	1
TTP	1	1	1	0

因此程序时间复杂度为：

$$T(n) = 3 \times O(\,(\log k) * k^2\,) + 2 \times O(k^2) + 1 \times O(k^3) + 1$$
$$= O(k^3) + (3\log k + 2) \cdot O(k^2) + 1$$

空间复杂度：

各端参数表中有 2^{32} 位数 4 个，2^{1024} 位数 8 个

公共参数中有 2^{32} 位数 1 个，2^{1024} 位数 4 个

使用中间变量 2^{1024} 位数 6 个

因此，变量所占空间复杂度：

$$S(n) = 18 \times 128Bit + 5 \times 4Bit = 2324Bit$$

7 场景三

7.1 参数表

场景三的游戏解决方案中，TTP、Alice、Bob 参数表分别如下表 7.1、表 7.2、表 7.3 所示，使用到的 Basic 库参数参见 4.1 节公有参数表。

表 7.1 场景三 TTP 参数表

名字	含义	位数
key	本方私钥	2^{32}
public_key	本方公钥	2^{1024}
rAC	Alice、TTP 约定参数	
rBC	Bob、TTP 约定参数	
r3	三方约定参数	

表 7.2 场景三 Alice 参数表

名字	含义	位数
grade	Alice 的成绩	2^{32}
rAB	Alice、Bob 约定随机数	2^{1024}
rAC	Alice、TTP 约定参数	
r3	三方约定参数	

表 7.3 场景三 Bob 参数表

名字	含义	位数
grade	Bob 的成绩	2^{32}
rAB	Alice、Bob 约定随机数	2^{1024}
rBC	Bob、TTP 约定参数	
r3	三方约定参数	

7.2 程序交互过程图

场景三程序交互过程如下图 7 所示。图中“(n)”形式的标号为程序中步骤执行顺序，各步骤含义如下：

(1) $(g^{r_{AB}})^{S_A}+r_{AC}$：Alice 使用约定参数 *rAB* 和自己的成绩计算得到 Pedersen 承诺，将该承诺再加上自己与 TTP 的约定参数，将此密文发送给 TTP

(1) $(g^{r_{AB}})^{S_B}+r_{BC}$：Bob 使用约定参数 *rAB* 和自己的成绩计算得到 Pedersen 承诺，将该承诺再加上自己与 TTP 的约定参数，将此密文发送给 TTP

(2) TTP 将 Alice 发送信息减去 Alice 与自己的约定参数，将 Bob 发送信息减去 Bob 与自己约定的参数，相减后再用自己的公钥进行加密，将加密结果再加上约定参数 *r3* 求得 M。

(3)TTP 再将 *M* 发送给 Alice 和 Bob

(4)Alice 和 Bob 分别各自将 TTP 发送的 M 与约定参数 *r3* 进行比较。若相同，则 *M* 的第一项为 0，即二人的 Pedersen 承诺值相同，所以二人成绩相同；否则表示二人成绩不同

没有人作弊时，Alice 和 Bob 均可得到正确结论；而只要出现作弊，二者均得到“对方成绩与自己成绩不同”的错误结论。

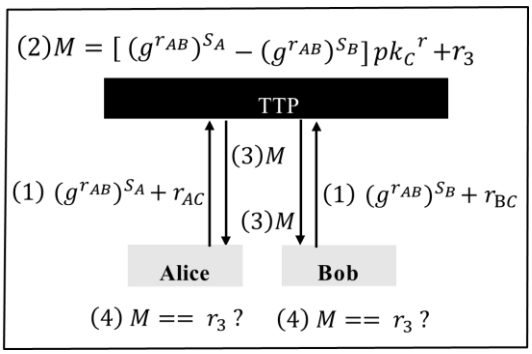

图 7 场景三程序交互过程图

7.3 伪代码

程序主要分为初始化阶段、游戏阶段 2 个阶段。

1、初始化阶段：

初始化阶段完成表 7.1、表 7.2、表 7.3 中的 TTP、Alice、Bob 参数表参数的初始化。

各端伪代码如下文算法所示，TTP 的初始化阶段函数伪代码为算法 7.1.1，Alice 的初始化阶段函数伪代码为算法 7.1.2，Bob 的初始化阶段函数伪代码为算法 7.1.3。

算法 7.1.1. TTP-Init

输入：无

输出：初始化 TTP 的参数表

Creat_Key(key, public_key);

mpz_urandomm(rAC, seed, p);

mpz_urandomm(rBC, seed, p);

mpz_urandomm(r3, seed, p);

Send_Agreement_Number("Third", "Alice", *rAC*);

Send_Agreement_Number("Third", "Alice", *r3*);

Send_Agreement_Number("Third", "Bob", *rBC*);

Send_Agreement_Number("Third", "Bob", *r3*);

说明：

1. TTP 的初始化首先生成一对自己的密钥对

2. 然后产生约定参数 *rAC*、*rBC*、*r3*
3. 使用 *Send_Agreement_Number()*将约定参数 *rAC*、*r3* 发送给 Alice；将约定参数 *rBC*、*r3* 发送给 Bob。

算法 7.1.2.　Alice-Init

输入：已经初始化好的S_A

输出：初始化 Alice 的参数表

```
mpz_urandomm(rAB, seed, p);
Send_Agreement_Number("Alice", "Bob", rAB);
Recieve_Agreement_Number("Alice", "Third", rAC);
Recieve_Agreement_Number("Alice", "Third", r3);
```

说明：

1. Alice 的初始化首先产生约定参数 *rAB*，使用 *Send_Agreement_Number()*将约定参数 *rAB* 发送给 Bob。
2. 然后使用 *Recieve_Agreement_Number()*从 TTP 接收约定参数 *rAC*、*r3*。

算法 7.1.3.　Bob-Init

输入：已经初始化好的S_B

输出：初始化 Bob 的参数表

```
Recieve_Agreement_Number("Bob", "Alice", rAB);
Recieve_Agreement_Number("Bob", "Third", rBC);
Recieve_Agreement_Number("Bob", "Third", r3);
```

说明：

1. 首先使用 *Recieve_Agreement_Number()*从 Alice 获取约定参数 *rAB*。
2. 然后使用 *Recieve_Agreement_Number()*从 TTP 接收约定参数 *rBC*、*r3*。

2、游戏阶段：

游戏阶段用程序实现图 7 中的交互过程。

各端伪代码如下文算法所示，TTP 的游戏阶段函数伪代码为算法 7.2.1，Alice 的游戏阶段函数伪代码为算法 7.2.2，Bob 的游戏阶段函数伪代码为算法 7.2.3。

算法 7.2.1.　TTP-Game

输入：无

输出：完成场景三的游戏过程

```
WHILE (1)
{   IF (Menu() == FALSE) BREAK;
    Init();
    WHILE (Recive_Message("Alice_Send.txt", recive1) == FALSE)
        Sleep(1000);
    mpz_sub(commit1, recive1, rAC);
    WHILE (Recive_Message("Bob_Send.txt", recive2) == FALSE)
        Sleep(1000);
    mpz_sub(commit2, recive2, rBC);
    mpz_sub(send, commit1, commit2);
    ElGamal_Lock(Random(), public_key, send, temp);
    mpz_add(send, temp, r3);
    Send_Message("Alice\\Third_Send.txt", send);
    Send_Message("Bob\\Third_Send.txt", send);
}
```

说明：

1. 最外层的 while (1)配合 *Menu()*菜单，实现 *Menu()*返回 false 即选择“退出游戏”时终止程序（即跳出该循环）的目的。
2. TTP 的一次游戏中，首先 *Menu()*选择是否退出游戏。
3. 然后调用 *Init()*函数完成初始化阶段
4. 然后等待接收 Alice 和 Bob 发送的密文
5. 将 Alice 发送信息减去 Alice 与自己的约定参数得到 Alice 的承诺，存储在 *commit1*
6. 将 Bob 发送信息减去 Bob 与自己约定的参数得到 Bob 的承诺，存储在 *commit2*
7. *commit1* 减去 *commit2*，再用自己的公钥进行加密，将加密结果再加上约定参数 r3，存储在 *send*。
8. 最后将 *send* 发送给 Alice 和 Bob。

算法 7.2.2.　Alice-Game

输入：无

输出：完成场景三的游戏过程

```
WHILE (1)
{   IF (Menu() == FALSE) BREAK;
    Init();
    IF (Choose() == TRUE)
    {
        mpz_powm(temp, Pedersen_g, rAB, p);
        mpz_powm_ui(commit, Pedersen_g, grade, p);
        mpz_add(send, commit, rAC);
    }
```

```
    ELSE
        mpz_urandomm(send, seed, p);
    Send_Message("Third\\Alice_Send.txt", send);
    WHILE (Recive_Message("Third_Send.txt", recive) == FALSE)
        Sleep(1000);
    IF (r3== recive) {成绩相同界面; }
    ELSE {显示成绩不同界面; }
}
```

说明：

1. 最外层的 while (1)配合 *Menu()*菜单，实现 *Menu()*返回 false 即选择“退出游戏”时终止程序（即跳出该循环）的目的。
2. Alice 的一次游戏中，首先 *Menu()*选择其成绩或是选择退出游戏。
3. 然后调用 *Init()*函数完成初始化阶段
4. 然后使用约定参数 *rAB* 与自己的成绩计算 Pedersen 承诺，再加上自己与 TTP 的约定参数 *rAC*
5. 然后 *Choose()*选择发送真实信息、还是使用随机数生成函数随机生成数字进行发送，选择后发送相应信息
6. 然后等待 TTP 发送的信息
7. 将收到信息与约定参数 *r3* 进行比较，相同成绩相同，显示二人成绩相同的界面，不同则显示二人成绩不同的界面。

算法 7.2.3. Bob-Game

输入：无

输出：完成场景三的游戏过程

```
WHILE (1)
{   IF (Menu() == FALSE) BREAK;
    Init();
    IF (Choose() == TRUE)
    {
        mpz_powm(temp, Pedersen_g, rAB, p);
        mpz_powm_ui(commit, Pedersen_g, grade, p);
        mpz_add(send, commit, rBC);
    }
    ELSE
        mpz_urandomm(send, seed, p);
    Send_Message("Third\\Bob_Send.txt", send);
    WHILE (Recive_Message("Third_Send.txt", recive) == FALSE)
        Sleep(1000);
    IF (r3== recive) {成绩相同界面; }
    ELSE {显示成绩不同界面; }
}
```

说明：

1. 最外层的 while (1)配合 *Menu()*菜单，实现 *Menu()*返回 false 即选择“退出游戏”时终止程序（即跳出该循环）的目的。
2. Bob 的一次游戏中，首先 *Menu()*选择其成绩或是选择退出游戏。
3. 然后调用 *Init()*函数完成初始化阶段
8. 然后使用约定参数 *rAB* 与自己的成绩计算 Pedersen 承诺，再加上自己与 TTP 的约定参数 *rBC*
4. 然后 *Choose()*选择发送真实信息、还是使用随机数生成函数随机生成数字进行发送，选择后发送相应信息
5. 然后等待 TTP 发送的信息
6. 将收到信息与约定参数 *r3* 进行比较，相同成绩相同，显示二人成绩相同的界面，不同则显示二人成绩不同的界面。

7.4 安全性证明

性质 1：假设 Alice 和 Bob 不会篡改自己的成绩，那么他们在游戏中一定能得到正确的判断结果，并且无法知道对方成绩。

证明.

Alice、Bob 无人作弊时，TTP 可通过 Alice 发送的信息 $(g^{r_{AB}})^{S_A}+r_{AC}$计算出 Alice 承诺值的真实值 $(g^{r_{AB}})^{S_A}$；可通过 Bob 发送的信息 $(g^{r_{AB}})^{S_B}+r_{BC}$计算出 Bob 承诺值的真实值 $(g^{r_{AB}})^{S_B}$。

因此对于 Alice 和 Bob 端接收到的信息为：

Alice：$M=[((g^{r_{AB}})^{S_A}-(g^{r_{AB}})^{S_B})]pk_c^r+r_3$

Bob：$M=[((g^{r_{AB}})^{S_A}-(g^{r_{AB}})^{S_B})]pk_c^r+r_3$

因此有以下两种情况：

Alice、Bob 成绩相同时：Alice 和 Bob 各自的承诺值 $(g^{r_{AB}})^{S_A}==(g^{r_{AB}})^{S_B}$，因此 M 第一项 $[((g^{r_{AB}})^{S_A}-(g^{r_{AB}})^{S_B})]pk_c^r==0$，即$M=r_3$。此时得到正确的判断结果。

Alice、Bob 成绩不同时：Alice 和 Bob 各自的承诺值 $(g^{r_{AB}})^{S_A}!=(g^{r_{AB}})^{S_B}$，因此 M 第一项 $[((g^{r_{AB}})^{S_A}-(g^{r_{AB}})^{S_B})]pk_c^r!=0$，即$M!=r_3$。此时

得到正确的判断结果。

并且，Alice 和 Bob 没有获得任何对方发送的信息，且 Alice 和 Bob 没有第三方公钥所对应的私钥，此时由定理 2 知，$[(\ (g^{r_{AB}})^{S_A} - \ (g^{r_{AB}})^{S_B})]pk_c^r$无法被 Alice、Bob 破解，即无法从第三方发送的 *M* 中获取对方成绩，即满足双方无法知道对方成绩。

综上所述：假设 Alice 和 Bob 不会篡改自己的成绩，那么他们在游戏中一定能得到正确的判断结果，并且无法知道对方成绩。

证毕.

性质 4：除了 Alice 和 Bob 之外的其他人不会在游戏中确定任何一个人的成绩。

证明.

对于第三方，其可获得信息为：

$$(g^{r_{AB}})^{S_A}、\ (g^{r_{AB}})^{S_B}$$

因第三方缺失 *rAB*，由定理 4 知：第三方无法由承诺值获取 Alice 和 Bob 成绩的真实值，但其可通过比较双方承诺是否相同，获取 Alice、Bob 成绩是否相同或不同。第三方端满足性质 4。

若存在监听方，则其可获得所有游戏中以文件形式发送的信息。则在游戏中其可获得的信息有：

$$(g^{r_{AB}})^{S_A} + r_{AC}、\ (g^{r_{AB}})^{S_B} + r_{BC}、$$
$$M = [(\ (g^{r_{AB}})^{S_A} - \ (g^{r_{AB}})^{S_B})]pk_c^r + r_3$$

因其缺失 *rAB*、*rAC*、*rBC*，由定理 4 及定义 1 知，其无法通过 $(g^{r_{AB}})^{S_A} + r_{AC}$、$(g^{r_{AB}})^{S_B} + r_{BC}$获取任何成绩有关信息。因其缺失 *r3* 以及第三方公钥所对应的私钥，由定理 2 及定义 1 知，监听方无法通过 *M* 计算出$[(\ (g^{r_{AB}})^{S_A} - \ (g^{r_{AB}})^{S_B})]$的值，即无法判断$[(\ (g^{r_{AB}})^{S_A} - \ (g^{r_{AB}})^{S_B})]pk_c^r == 0$是否成立，即无法通过 *M* 判断 Alice、Bob 成绩是否相同，且获取任何成绩有关信息。

综上所述：除了 Alice 和 Bob 之外的其他人不会在游戏中确定任何一个人的成绩。

证毕.

7.5 性能分析

时间复杂度：

各端游戏中使用到的操作如下表 T3 所示。其中模幂运算中的指数在场景三中为 2^{1024} 位数。

表 T2 场景二操作次数统计

	模幂	乘模	乘逆	随机数
Alice	2	0	0	0
Bob	2	0	0	0
TTP	1	1	0	0

因此程序时间复杂度为：

$$T(n) = 5 \times O(\ (\log k) * k^2\ ) + 1 \times O(k^2)$$
$$= (5\log k + 1) \cdot O(k^2)$$

空间复杂度：

各端参数表中有 2^{32} 位数 3 个，2^{1024} 位数 10 个

公共参数中有 2^{32} 位数 1 个，2^{1024} 位数 5 个

使用中间变量 2^{1024} 位数 8 个

因此，变量所占空间复杂度：

$$S(n) = 23 \times 128Bit + 4 \times 4Bit = 2960Bit$$