Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет имени
Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Робототехники и комплексной автоматизации»
КАФЕДРА «Системы автоматизированного проектирования (РК-6)»

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

по дисциплине «Защита информации»

Студент:	Александров Максим Алексеевич
Группа:	РК6-81Б
Тип задания:	Лабораторная работа
Тема:	2

Студент _____ (подпись, дата) Александров М.А. (Фамилия, И.О.)

Преподаватель _____ (подпись, дата) Беломойцев Д.Е. (Фамилия, И.О.)

Москва, 2025

Содержание

2

Задание

1. Сгенерировать пару ключей RSA; Выполнить шифрование произвольного набора данных; Выполнить расшифрование произвольного набора данных
2. Осуществить генерацию запроса на сертификат; Осуществить выпуск самоподписанного сертификата на данный запрос
3. Рассчитать хеш по алгоритму SHA для произвольного набора данных
4. Выпустить ЭЦП для произвольного набора данных на базе ключей и сертификатов из работы 2
5. Необходимо составить программную реализацию алгоритма генерации общего секретного ключа (алгоритм Диффи-Хеллмана)
6. Необходимо составить программную реализацию алгоритма шифрования RSA

1 Работа с RSA

```
all: first second third
  @echo "First task is complete!"

# ----------------------------------------

FILE          = file.txt
PRIVATE_KEY       = private_key.pem
PASSWORD        = my_password
PUBLIC_KEY        = public_key.pem

first:
  @echo "-----------------------------------"
  @echo "1.1 Generating key, based on $(FILE), password:
     $(PASSWORD)"
  @openssl genrsa \
  -out $(PRIVATE_KEY) \
  -des3 \
  -rand $(FILE) \
  -passout pass:$(PASSWORD) \
  4096

  @echo "Generating public key, based on secret key"
  @openssl rsa \
  -in $(PRIVATE_KEY) \
  -out $(PUBLIC_KEY) \
  -pubout \
  -passin pass:$(PASSWORD)

# ----------------------------------------

FILE_TO_ENCRYPT_DES3  = file_to_encrypt_des3.txt
ENCRYPTED_FILE_DES3   = encrypted_file_des3.txt

FILE_TO_ENCRYPT_RSA   = file_to_encrypt_rsa.txt
ENCRYPTED_FILE_RSA    = encrypted_file_rsa.txt

second:
  @echo "-----------------------------------"
  @echo "1.2 Ecrypting file $(FILE_TO_ENCRYPT_DES3) to
     $(ENCRYPTED_FILE_DES3)"
  @openssl des3 \
  -in $(FILE_TO_ENCRYPT_DES3) \
  -out $(ENCRYPTED_FILE_DES3) \
  -pass pass:$(PASSWORD)
```

```
43
44    @echo "Encrypting file $(FILE_TO_ENCRYPT_RSA) to
         $(ENCRYPTED_FILE_RSA)"
45    @openssl rsautl \
46    -in $(FILE_TO_ENCRYPT_RSA) \
47    -out $(ENCRYPTED_FILE_RSA) \
48    -inkey $(PUBLIC_KEY) \
49    -pubin -encrypt
50
51 # -----------------------------------------
52
53 DECRYPTED_FILE_DES3   = decrypted_file_des3.txt
54 DECRYPTED_FILE_RSA    = decrypted_file_rsa.txt
55
56 third:
57    @echo "---------------------------------"
58    @echo "1.3 Decrypting file $(ENCRYPTED_FILE_DES3) to
         $(DECRYPTED_FILE_DES3)"
59    @openssl des3 -d \
60    -in $(ENCRYPTED_FILE_DES3) \
61    -out $(DECRYPTED_FILE_DES3) \
62    -pass pass:$(PASSWORD)
63
64    @echo "Decrypting file $(ENCRYPTED_FILE_RSA) to
         $(DECRYPTED_FILE_RSA)"
65    @openssl rsautl \
66    -in $(ENCRYPTED_FILE_RSA) \
67    -out $(DECRYPTED_FILE_RSA) \
68    -inkey $(PRIVATE_KEY) \
69    -passin pass:$(PASSWORD) \
70    -decrypt
71
72 # -----------------------------------------
73
74 clean:
75    @rm -rf $(PRIVATE_KEY) $(PUBLIC_KEY) \
76    $(ENCRYPTED_FILE_DES3) $(DECRYPTED_FILE_DES3) \
77    $(ENCRYPTED_FILE_RSA) $(DECRYPTED_FILE_RSA)
```

execution:

```
1 make first
2 ==================================
3 make[1]: Entering directory
     '/home/maxim/study/information_security/1
4 '
```

```
   ---------------------------------
1.1 Generating key, based on file.txt, password: my_password
Generating public key, based on secret key
writing RSA key
---------------------------------
1.2 Ecrypting file file_to_encrypt_des3.txt to
   encrypted_file_des3.tx
t

   *** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
Encrypting file file_to_encrypt_rsa.txt to
   encrypted_file_rsa.txt
The command rsautl was deprecated in version 3.0. Use 'pkeyutl'
   inste
ad.

   ---------------------------------
1.3 Decrypting file encrypted_file_des3.txt to
   decrypted_file_des3.tx
t

   *** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
Decrypting file encrypted_file_rsa.txt to decrypted_file_rsa.txt
The command rsautl was deprecated in version 3.0. Use 'pkeyutl'
   inste
ad.

   First task is complete!
make[1]: Leaving directory
   '/home/maxim/study/information_security/1'
```

file.txt:

```
File, that used in ecryption
```

file_to_encrypt_rsa.txt:

```
Another secret information
```

encrypted_file_rsa.txt содержит нечитаемые бинарные данные
decrypted_file_rsa.txt:

```
Another secret information
```

public_key.pem:

```
-----BEGIN PUBLIC KEY-----
```

```
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAuTe3W9jll1yxKFu8r292
jqY30BA5AHKWUpGqI+rrwiaEHOqahSHVcxt45L3fUmPu74eg/NflphMVaAMFiF/2
1W00Q5OIaYuFGJ6al/7q3InTggIuwLgVwEeEw8aAFp5HyJn/22ve/Lbw/rxBEzY4
vv8utR/YTUsHxDFaoNpGCmZh0PRXbLvIOIc/LQMSCmpUlsnqz7KYkNd1ZNjh0V8C
8RIhACyvi2DueOCD7IhUFwoql0exu8vxAb7C0eI/1rPSiY5fStBqiu2m45oU0WbV
tat3AomM5XkNQbiHcuqa/UUE0xGFiTK7u7LRqLpb2Hf+AgTokpPYIvFXFemqQrcc
GIuuJRp7xQ7Uhcbw7IBR+nxCbfydGTgfYHRuIi8TmaS3VwjLW2ziZfkETah7t8nq
//3UxLGxusMnuMaZWHWjt9ZxcEvdVnZoAPzlmYs+pjAMEMGKwxzSoaqvZ2CenGKn
ftaWURAOPUnVigCxFHkbTOTCeWKSRlwBai14kvi2vUKfsbJjAyg1y8q4O6bTYsVU
PrdENmBlS4iS3m8aHlzcYlXbclovFzIlNzhstGDwcQpZbLKoU/n06Vf8zVJ2Q/vu
jaMmP6/6Hfytw0RP6kQhzNm8Pya/uZ6/xwXhk7TeHlnaanbac7eHab3n8ilrCzum
TUBk95NTxrmID6gyV/EcRPMCAwEAAQ==
-----END PUBLIC KEY-----
```

private_key.pem:

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIJnDBOBgkqhkiG9w0BBQ0wQTApBgkqhkiG9w0BBQwwHAQIWYQ1zMuNOnYCAggA
MAwGCCqGSIb3DQIJBQAwFAYIKoZIhvcNAwcECNJkStg7/RHKBIIJSJXsA2E/Zfji
j1QmEhQeVik9llzkWRnI9WB1vnw63L85eqd9ObFCPBrN7Y7zr2dkvE9tD74/f0Kt
my3L327vCENKN+K4GjJg+d4fbxWzBUoplputdfw3TLLFY3D4fboacZQk28mo/Klc
+ee0su4nxDg2sTrboJGr2e04OAXsz7OqP9IvSE0V3fUKmH8tEQeYDkuxuEiTmUvy
fsaV09JymntUYeAfRr24VypLZk20phEOUG//eMvfoGXCr3HWG+fMYStfQNVgvWAs
/H8YONS+UWRu65DgQmLD+NHkDZzZAsfr6nqJLCmXUF1ZAsKvb2xSxwlVUe/oZNrZ
F7NszfO7cQBanEkzbcZNpITsZoDbBeVPzyDRV+JoJt9yP5Zi1w6DD/eSN/Jv6CCb
ak4aa9Jlrrb5HWZqXTrM1E2TaPFVVRF4Dm4aEUmlZbTYFdzf72oeBeonoAa26ao3
CC3vOBkzCVteFpfUocmjEhM2buRkuC5cX3Kwg5sJ7rvozmMh+IN/uLfH/2SqCsm5
HppMoBuvrHldtyYasbupE5Y9SDtKDk0RtjaCJtKAZEF1ngfAB43BDwXzhMNiaVa8
cQdIJWMAp/oqnUYBfyDnsFm1oYB2NpYF8ktbeOeUeQFYOX3tx3KDbSs0YhpjVmU+
PtoK4ZXfSAH4361BxYrp2aymbqCnMpr6iF/HNqFaFQDaIDYiisFvoMLYABIFtZgZ
OkpPck+j4Ha4T820+MDIrl5zKwtNKsNhjtBbIdy3cWABAOIMHHkswv5Z2nj06sAu
3Aca1BNnV0khZOKLqHzecJ5cwIBVtjvZ+gl/XKwcHXy72BQRNqlOUb7uzRd4vtlQ
AfiMvNOsIzuZtZzvVtP4emeuWozPU5bGTh7C0w6sEotdMyQqAvDoiWtxtFk4SGxu
4Py/j7T/GainidirkwSahTTSqqkGei7V/2DAdpQod9tTBnVJlwxaTZXMLGwjtgqP
BbOb/i+HGSIf/1sB7v2ENn1/EB6VzWrAx8t4yKDxsdCPQQF8NxEI7zO505FDYK/m
6Kaid8fxOJIMg9sGRKG2CSwFi9sZsl59nr/PVqJwdcVJRJAQQxR21SvOTnIZFZQA
q1UZussXaJ1lLI7lHrYP4PN2uApRiD6nMGCZ2AePu8wjvjydrR5xu5OHbiT8mbON
EgYyJbrFlXEwTMYowwgUrGfRskLXPGwwjNHwBLW4q9RHDkGmcHsisi6LCIqybrhQ
wioJ0DMu9BbVtjhyCeLcfeKKnjgJdIdPu21WIrHcJYDub222VNtOSyAtkYispuKa
TRXSAyULq8UMdaKS3GFIuQqOtEtKyzGUvtUZ9YFN21FJlf5myvmcs3m9o7yANtzC
g6PsREWdrzmvZvDxpNDNZ/mY+My7+2iqpOoNzSfNkHPBFXaLXGyLsEpC8RinV+Mm
6k1C0unClPBv1i2ce4lFrKdudYH+P/Hx4xXulbwn7bYO/xOXU0OAmhvOGxl5r8Sj
egkWnR3WiniFvp7WH4WiLJ0DbVPRtC4RsSkMs8PKHHz+5I2J0eCk2876Pkhwciaz
RAZcco0seCEUE3fudrrZkYDdn/mQXMsDIvaey1kmrYPdlJLmeqQHrGC4rA9WEo5w
w9H6Fnckn4sECaYwRQVljDeb1QDo6uwzm3XoxZVUb88RYeUGqRjuWnV3ENH4eMom
Co3O6u3XxtUPaBU1GItnG6WYwvwmPdR985bzAsmnfLLl+JxM8tChN1dMu3u/5n2c
FlnWT4cpzlMdqMTvHkJgcEnESii8ZP0QEXtApbNr0Fkcl7NNA1u5dEyiygDFJUZS
```

```
jsn9YU5E3DhHagEefEHXhrDzNip01wG0tZwLT1lBREC0Ickl2oSXG6Qgj6cvAvAf
Vda70JkL04l1c/XZZMphyOalfisy08GE7MhrPedcRHimBDESV2q5nDteqcCx6cwO
diHNhZbh0A08SVDWJI4p9yo/3d2HpTegCTGlekE3e9Pzv9SLsPEBuwycz+AyLFcF
WcDNRkEanHgThjMMn1oDK6EgF3I1YOKDHWDX2HAmM+tq4QwRqGDvSWV03TQmGwhi
T2s3Jw/WbdCKgSp6RzQJ69AuLah6sXdZUaSJGl5XJM2dGXjodN7fLhGRSVGtC3NW
LtSlciGwdwZcbcdA2E9Kb+DL1OCjzJLSOOtH9xmp7jOF1Yp3kOo3wbBcY+sLkbAW
/4oqDm1xqMKW02FeIB6aQdeqQWEIcm5fblWRUIOuh/KWM8TVgi45acOnLeb+0v0F
AKDg1Hbz4l94IdzXxvU/1c6NkFOdHIChUnOrrs17c8HiQtdE1z3/LPOlOWGtOWJe
JZ3RiM5MMfN23GmWZgaYDZ+DHetKGoNfrNi3Vl0EqRV+SbrDbCuyxhYvqadzT3i/
tjzzwroInbkHmMDExmPHkPbL2fZCGuhtbMe46AdPblM1EaWDSh19ekU06pHgqbby
EOcLvQ21AUcq5DEFbv9xEg0t9U5Ie/MiPZhLlMHv2mb0eDDO/F6CpsjxEZ3rhMqY
WKrzApmBq72fu6MzJV9l8WRTnoE2LpqnImxKXtAOIgsNeAjWGqetrEF/xznsg1UP
TS6SjxX1w6vMDdd8nL8T3kwduhuWUPmMPVPsbznVB47Va3KrO0EZp2+DBk0FB6dL
ENtiNlXDmKQNW6VJdkesMznA1NczPDt4vPp+I2nqsJU8vfSYbID4NvU4LDvjkjB2
gYc5fpN04qWXk2XqSccC/z6Zqmj3TVsaxaPpGeMnlIxL+CFBdwcrOlJ7xgS91MQz
HxyolIngf8JQGCaAZ1Ss6jCH+VtEINHNbm+NihHGYYb3L4gfHvEvnQQFVolJW5JE
Hbt/sgPp5uxq9I9O6ESFMKT0HB8yjQTPsglygA7l2TCMxKgdC/MUi9YX7VkuKyfv
8+obLYfWnFVNGY73PNTejY0dfdAt8rMj9cMMaQbLX71rsCzcEQiYdvev2sWWmXmb
Rzhx7Hv5nfVzj7pOO7SHx7I6nFg0bsRjcDpvDn2SEAFRLFFaolkizwDT3RbBPXf/
RpAJkt8CuMN9QwXD6yaoff4OWuwL2aYyDIjDXQui0OktJZX2Y4eqYlR1VlWWtTJl
uryXgGfwPukkDoKlsLFhLHhX5wcYPrutQuBAFxZ0OuV3x0KiCLgKg0OTw/h5xgPZ
Uc1BONb4zUd+4BPPfssCFw==
-----END ENCRYPTED PRIVATE KEY-----
```

2 Работа с сертификатами

```
all: first
  @echo "Second task is complete!"

FILE          = file.txt
PASSWORD        = my_password
PRIVATE_KEY       = private_key.pem
PUBLIC_KEY        = public_key.pem
FILE_TO_SIGN      = file_to_sign.txt
HASH_FILE       = hash_file.txt
SIGNATURE       = file_to_sign.sig
SERTIFICATE       = sertificate.csr
SELF_SIGNED_CERT    = self_signed_cert.csr

first:
  @echo "----------------------------------"
  @echo "2. Signing file"

  @echo "................................."
  @echo "Generating private key"
  @openssl genrsa \
  -out $(PRIVATE_KEY) \
  -des3 \
  -rand $(FILE) \
  -passout pass:$(PASSWORD) \
  4096

  @echo "................................."
  @echo "Generating public key"
  @openssl rsa \
  -in $(PRIVATE_KEY) \
  -out $(PUBLIC_KEY) \
  -pubout \
  -passin pass:$(PASSWORD)

  @echo "................................."
  @echo "Counting hash"
  @openssl dgst \
  -sha256 \
  -out $(HASH_FILE) \
  $(FILE_TO_SIGN)

  @echo "................................."
  @echo "Creating signature"
  @openssl dgst \
```

```
45    -sha256 \
46    -sign $(PRIVATE_KEY) \
47    -out $(SIGNATURE) \
48    -passin pass:$(PASSWORD) \
49    $(FILE_TO_SIGN)
50
51    @echo "................................"
52    @echo "Verifying signature"
53    @openssl dgst \
54    -sha256 \
55    -verify $(PUBLIC_KEY) \
56    -signature $(SIGNATURE) \
57    -passin pass:$(PASSWORD) \
58    $(FILE_TO_SIGN)
59
60    @echo "................................"
61    @echo "Generating CSR"
62    @openssl req \
63    -new \
64    -key $(PRIVATE_KEY) \
65    -out $(SERTIFICATE) \
66    -passin pass:$(PASSWORD) \
67    -subj "/C=RU/ST=MO/L=Moscow/O=BMSTU/OU=RK6/CN=Certificate"
68
69    @echo "................................"
70    @echo "Generating self-signed certificate"
71    @openssl x509 \
72    -req \
73    -in $(SERTIFICATE) \
74    -signkey $(PRIVATE_KEY) \
75    -out $(SELF_SIGNED_CERT) \
76    -days 1 \
77    -passin pass:$(PASSWORD)
78
79  clean:
80    @rm -rf $(PRIVATE_KEY) $(PUBLIC_KEY) $(HASH_FILE)
        $(SIGNATURE) $(SERTIFICATE) $(SELF_SIGNED_CERT)
```

execution:

```
1  make second
2  ==================================
3  make[1]: Entering directory
       '/home/maxim/study/information_security/2
4  '


       ----------------------------------
```

```
2. Signing file
.................................
Generating private key
.................................
Generating public key
writing RSA key
.................................
Counting hash
.................................
Creating signature
.................................
Verifying signature
Verified OK
.................................
Generating CSR
.................................
Generating self-signed certificate
Certificate request self-signature ok
subject=C = RU, ST = MO, L = Moscow, O = BMSTU, OU = RK6, CN =
   Certif
icate

   Second task is complete!
make[1]: Leaving directory
   '/home/maxim/study/information_security/2'
```

3 Хэширование

```
all: first
  @echo "Third task is complete!"

FILE_TO_COUNT_HASH    = file_to_count_hash.txt

first:
  @echo "---------------------------------"
  @echo "3. Countng hash of file $(FILE_TO_COUNT_HASH)"
  @openssl md5 -c $(FILE_TO_COUNT_HASH)
  @openssl sha1 $(FILE_TO_COUNT_HASH)

clean:
  @/bin/true
```

execution:

```
make third
==================================
make[1]: Entering directory
   '/home/maxim/study/information_security/3
'

   ---------------------------------
3. Countng hash of file file_to_count_hash.txt
MD5(file_to_count_hash.txt)=
   d4:1d:8c:d9:8f:00:b2:04:e9:80:09:98:ec:f
8:42:7e

   SHA1(file_to_count_hash.txt)=
   da39a3ee5e6b4b0d3255bfef95601890afd8070
9

   Third task is complete!
make[1]: Leaving directory
   '/home/maxim/study/information_security/3'
```

4 Выпуск ЭЦП

Выполнено во втором пункте командами:

```
@echo "................................."
@echo "Generating CSR"
@openssl req \
-new \
-key $(PRIVATE_KEY) \
-out $(SERTIFICATE) \
-passin pass:$(PASSWORD) \
-subj "/C=RU/ST=MO/L=Moscow/O=BMSTU/OU=RK6/CN=Certificate"

@echo "................................."
@echo "Generating self-signed certificate"
@openssl x509 \
-req \
-in $(SERTIFICATE) \
-signkey $(PRIVATE_KEY) \
-out $(SELF_SIGNED_CERT) \
-days 1 \
-passin pass:$(PASSWORD)
```

5 Реализация алгоритма генерации

Программа:

```
#include <iostream>
#include <openssl/bn.h>

class TBigNumber {
public:
    TBigNumber() : bn_(BN_new()) {
        if (!bn_) throw std::runtime_error("Failed to create
           BIGNUM");
    }

    ~TBigNumber() {
        if (bn_) {
            BN_free(bn_);
        }
    }

    BIGNUM* get() const {
        return bn_;
    }

    operator BIGNUM*() const {
        return bn_;
    }

private:
    BIGNUM* bn_;
};

class TBN_CTX {
public:
    TBN_CTX() : ctx_(BN_CTX_new()) {
        if (!ctx_) throw std::runtime_error("Failed to create
           BN");
    }

    ~TBN_CTX() {
        if (ctx_) {
            BN_CTX_free(ctx_);
        }
    }

    BN_CTX* get() const {
        return ctx_;
```

```
42      }
43
44      operator BN_CTX*() const {
45          return ctx_;
46      }
47
48  private:
49      BN_CTX* ctx_;
50  };
51
52  int main() {
53      try {
54          TBigNumber p;
55          TBigNumber g;
56          TBigNumber a;
57          TBigNumber b;
58          TBigNumber A;
59          TBigNumber B;
60          TBigNumber shared_key_A;
61          TBigNumber shared_key_B;
62
63          TBN_CTX ctx;
64
65          // Определяемпростоечисло   p иоснование  g
66          BN_set_word(p, 9001);  // Например, 9001
67          BN_set_word(g, 5);   // Например, 5
68
69          // Генерируемприватныеключи   a и b
70          BN_rand(a, 256, -1, 0);  // Приватныеключи  a и b
               могутбытьслучайнымичислами
71          BN_rand(b, 256, -1, 0);
72
73          // Вычисляемоткрытыеключи   A = g^a mod p и B = g^b mod p
74          BN_mod_exp(A, g, a, p, ctx);
75          BN_mod_exp(B, g, b, p, ctx);
76
77          // Вычисляемобщийсекретныйключ    (shared secret)
78          // Сторона A будетвычислять  (B^a mod p)
79          BN_mod_exp(shared_key_A, B, a, p, ctx);
80          // Сторона B будетвычислять  (A^b mod p)
81          BN_mod_exp(shared_key_B, A, b, p, ctx);
82
83          // Выводимобщийсекретныйключдляпроверки
84          std::cout << "Shared key (computed by A): " <<
               BN_bn2dec(shared_key_A.get()) << "\n";
85          std::cout << "Shared key (computed by B): " <<
```

```
            BN_bn2dec(shared_key_B.get()) << "\n";
    } catch (const std::exception& ex) {
        std::cerr << "Error: " << ex.what() << "\n";
        return 1;
    }
    return 0;
}
```

execution:

```
make fifth
==================================
make[1]: Entering directory
   '/home/maxim/study/information_security/5
'


   ----------------------------------
==> Configuring the project_template...
-- Configuring done (0.0s)
-- Generating done (0.0s)
-- Build files have been written to:
   /home/maxim/study/information_se
curity/5/build
                                                                  ==>
   Building the project_template...
gmake[2]: Entering directory
   '/home/maxim/study/information_security/
5/build'

   gmake[3]: Entering directory
   '/home/maxim/study/information_security/
5/build'

   gmake[4]: Entering directory
   '/home/maxim/study/information_security/
5/build'

   gmake[4]: Leaving directory
   '/home/maxim/study/information_security/5
/build'

   [100%] Built target project_template
gmake[3]: Leaving directory
   '/home/maxim/study/information_security/5
/build'

   gmake[2]: Leaving directory
```

```
   '/home/maxim/study/information_security/5
/build'

   ==> Running project_template
make[1]: Leaving directory
   '/home/maxim/study/information_security/5'
```

outputs:

```
Shared key (computed by A): 6994
Shared key (computed by B): 6994
```

6 Реализация RSA

Программная реализация:

```
#include <iostream>
#include <numeric>

namespace {

// Функциядлянахожденияобратногомодуля, основаннаянарасширенномалгоритмеЕвклида
int InverseMod(int e, int phi) {
    int x0 = 0;
    int x1 = 1;
    int phi0 = phi;
    int temp = 0;

    while (e > 1) {
        int div = e / phi;
        temp = phi;
        phi = e % phi;
        e = temp;
        temp = x0;
        x0 = x1 - div * x0;
        x1 = temp;
    }

    if (x1 < 0) {
        x1 += phi0;
    }

    return x1;
}

// Функциядлявозведениявстепеньпомодулю
int PowerMod(int base, int exp, int mod) {
    int result = 1;
    base = base % mod;
    while (exp > 0) {
        if (exp % 2 == 1) {
            result = (result * base) % mod;
        }
        exp = exp >> 1;
        base = (base * base) % mod;
    }
    return result;
}
```

```
43
44} // namespace
45
46int main() {
47    // Используемнебольшиепростыечисладлянаглядности
48    constexpr int p = 61; // Простоечисло
49    constexpr int q = 53; // Простоечисло
50
51    constexpr int n = p * q; // Модуль
52    constexpr int phi = (p - 1) * (q - 1);
53
54    constexpr int e = 17; // Публичнаяэкспонента ,
          котораявзаимнопростас    phi именьше  phi
55
56    // Проверяем, чтобы e и phi быливзаимнопростыми
57    static_assert(std::gcd(e, phi) == 1);
58
59    // Находимприватнуюэкспоненту   (d)
60    int d = InverseMod(e, phi);
61
62    // Выводимоткрытыйизакрытыйключи
63    std::cout << "Public Key: (" << e << ", " << n << ")\n";
64    std::cout << "Private Key: (" << d << ", " << n << ")\n";
65
66    // Шифрованиеидешифрование
67    int message = 42; // Этонашеисходноесообщение
68    int encrypted = PowerMod(message, e, n);
69    int decrypted = PowerMod(encrypted, d, n);
70
71    // Результаты
72    std::cout << "Original Message: " << message << "\n";
73    std::cout << "Encrypted Message: " << encrypted << "\n";
74    std::cout << "Decrypted Message: " << decrypted << "\n";
75
76    return 0;
77}
```

execution:

```
1make sixth
2==================================
3make[1]: Entering directory
    '/home/maxim/study/information_security/6
4'


    ----------------------------------
5==> Configuring the project_template...
```

```
-- Configuring done (0.0s)
-- Generating done (0.0s)
-- Build files have been written to:
   /home/maxim/study/information_se
curity/6/build
                                                                    ==>
   Building the project_template...
gmake[2]: Entering directory
   '/home/maxim/study/information_security/
6/build'

   gmake[3]: Entering directory
   '/home/maxim/study/information_security/
6/build'

   gmake[4]: Entering directory
   '/home/maxim/study/information_security/
6/build'

   gmake[4]: Leaving directory
   '/home/maxim/study/information_security/6
/build'

   [100%] Built target project_template
gmake[3]: Leaving directory
   '/home/maxim/study/information_security/6
/build'

   gmake[2]: Leaving directory
   '/home/maxim/study/information_security/6
/build'

   ==> Running project_template
make[1]: Leaving directory
   '/home/maxim/study/information_security/6'
```

output:

```
Public Key: (17, 3233)
Private Key: (2753, 3233)
Original Message: 42
Encrypted Message: 2557
Decrypted Message: 42
```