Лабораторная работа 4: «Создание и верификация цифровой подписи» [до 12 мая]

1. О деталях реализации и средствах разработки

- Генерация закрытого/секретного ключа (в формате `PEM`) для применения шифра RSA:

```
openssl genpkey -algorithm RSA -out privatekey.pem -pkeyopt
↪  rsa_keygen_bits:1024
```

  По умолчанию ключ содержит 4096 битов, произвольный размер устанавливается через `pkeyopt`.

- Формирование открытого/публичного ключа по закрытому ключу:

```
openssl rsa -pubout -in privatekey.pem -out publickey.pem
```

  Сгенерированный ключ можно просмотреть в любом текстовом редакторе.

- Вывод информации о структуре сгенерированного закрытого ключа (модуль, простые числа, экспоненты) и отдельно модуля:

```
openssl rsa -text -in privatekey.pem
```

```
openssl rsa -in privatekey.pem -noout -modulus
```

- Шифрование файла:

```
openssl rsautl -encrypt -inkey publickey.pem -pubin -in message.txt
↪  -out message.enc
```

- Дешифрование файла:

```
openssl rsautl -decrypt -inkey privatekey.pem -in message.enc -out
↪  message.dec
```

- Вычисление цифровой подписи файла:

```
openssl dgst -sha256 -sign privatekey.pem -out signature.bin
↪  message.txt
```

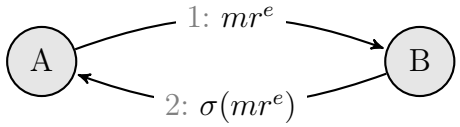
  Зашифровывается хешкод файла (команды можно разделить).

- Верификация цифровой подписи файла:

```
openssl dgst -sha256 -verify publickey.pem -signature signature.bin
↪  message.txt
```

  В случае неудачной верификации: «Verification Failure».

- Принципиальная схема слепой подписи[1] (все вычисления производятся в кольце вычетов по модулю n):



  Здесь: m – документ участника A, $r \in \mathbb{Z}_n \backslash \{0\}$ – известное только участнику A случайное число (маскирующий/затемняющий множитель), (e, n) – открытый ключ участника B. Участник B подписывает/заверяет документ mr^e:

$$\sigma(m \cdot r^e) = (m \cdot r^e)^d = m^d \cdot r^{ed} = m^d \cdot r = \sigma(m) \cdot r,$$

  где (d, n) – закрытый ключ участника B. Чтобы получить подпись участника B документа m, участник A должен умножить $\sigma(mr^e)$ на число, обратное r.

[1]Свойства слепой подписи: 1) нулевое разглашение – пользователь получает подпись на сообщении, не раскрывая самого сообщения подписывающей стороне; 2) непрослеживаемость – подписывающая сторона не может отследить пару подпись-сообщение после того, как пользователь обнародовал подпись на своем сообщении; 3) неподложность – только подписывающая сторона может сгенерировать действительную подпись. Ассоциация: если подписать конверт, в котором находится копировальный лист и документ под ним, то при вскрытии конверта документ также окажется подписанным.

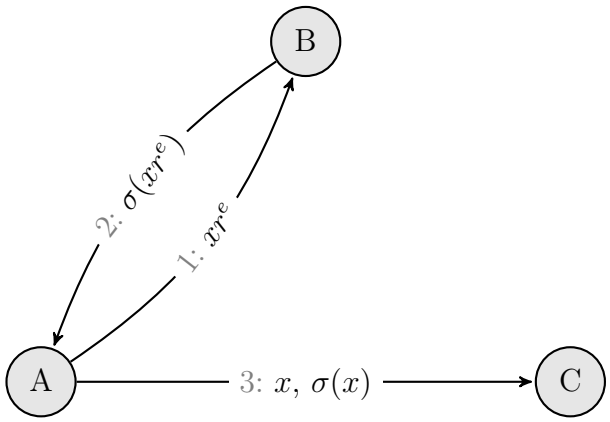
- Реализация алгоритма быстрого возведения в степень в кольце вычетов ($a^b \bmod n$):

```
long binpow(int a, int b, int n){
    long res = 1;

    while(b > 0){
        if((b&1) == 1)
            res = (res*a)%n;

        a = (a*a)%n;
        b >>= 1;
    }

    return res;
}
```

- Для вычисления обратного элемента в кольце классов вычетов используют расширенный алгоритм Евклида, т.к. НОД$(r, r^{-1}) = 1$.

2. Постановка задачи

Реализуйте простое клиент-серверное приложение, позволяющее аккумулировать короткие анонимные сообщения (систему электронного голосования) согласно следующей схеме:

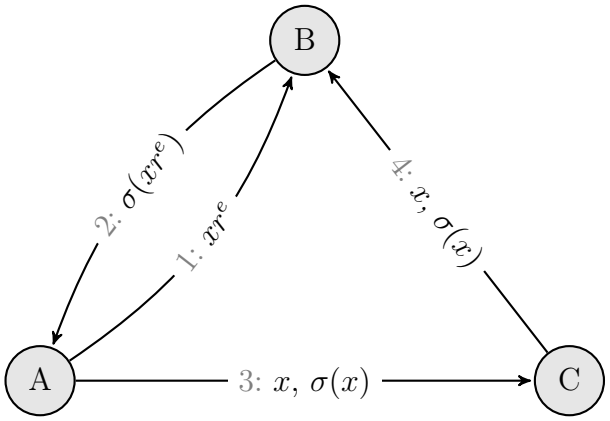


Здесь: A – пользователь (избиратель), B – регистратор, C – счетчик, x – сообщение (голос), r – известное только участнику A случайное число, (e, n) – открытый ключ банка. Пренебрегите реализацией правильных механизмов распределения, хранения и сертификации ключей.

3. Задания для подготовки к экзамену

1. Распишите (на примере) процедуру применения шифра RSA для шифрования и расшифрования одного сообщения. Входные данные: $p = 11$, $q = 7$, $e = 37$, открытое сообщение – 15. Повторите процедуру, самостоятельно подобрав значение экспоненте e.

2. Реализуйте простое клиент-серверное приложение, позволяющее реализовывать (тратить) бонусные/рейтинговые баллы (электронные деньги) согласно следующей схеме [обеспечение неотслеживаемости покупателя]:



Здесь: A – компания (банк), B – клиент, C – продавец, x – баллы (купюра: номинал и номер), e – открытый ключ банка.

Шаг 1: Покупатель производит купюру (генерирует серийный номер купюры x, включающий ее номинал) и затемнив отправляет ее банку; в общем случае сообщение клиента имеет вид

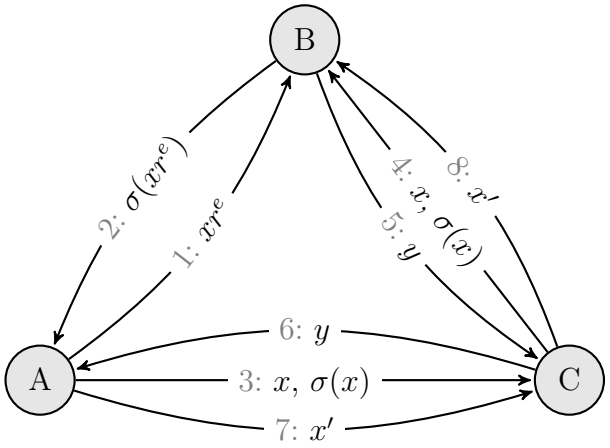
$$(\text{withdrawal}, \text{an}_a, xr^e)^{d_a},$$

где withdrawal – операция снятия денег со счета, an_a – номер счета клиента, d_a – закрытый ключ клиента. **Шаг 2:** Банк заверяет купюру, возвращает клиенту и снимает со счета клиента указанную сумму денег. **Шаг 3:** Клиент снимает затемняющий множитель, проверяет $(x^d)^e = x$ и отправляет купюру продавцу. **Шаг 4:** Продавец отправляет анонимную купюру банку, банк переводит деньги на счет продавца и изымает купюру из обращения (запоминает номер); в общем случае сообщение продавца имеет вид

$$(\text{deposit}, an_b, x^d)^{d_b},$$

где deposit – операция перевода денег на счет, an_b – номер счета продавца, d – закрытый ключ банка, d_b – закрытый ключ продавца; банк может отправить расписку продавцу вида $(\text{deposit}, an_b, x^d)^d$.

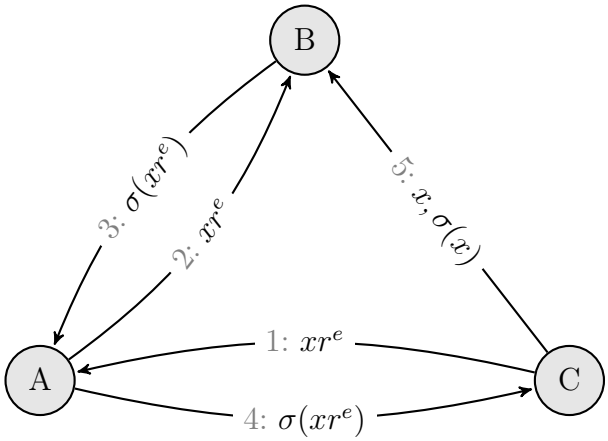
3. Реализуйте простое клиент-серверное приложение, позволяющее реализовывать электронные деньги согласно следующей схеме [усиление защиты через подтверждение аутентичности купюры]:



Здесь: A – банк, B – клиент, C – продавец, $x = h(x')$ – купюра, e – открытый ключ банка, r – известное только клиенту случайное число, y – подтверждение нахождения купюры в обращении.

Шаг 5: Банк отправляет продавцу зашифрованное на своем закрытом ключе заверение в законности сделки, если после проверки не обнаруживает номер купюры в списке выведенных из обращения. **Шаг 6:** продавец пересылает заверение банка клиенту. **Шаг 7:** Клиент проверяет заверение на аутентичность и отправляет продавцу прекурсор x'. **Шаг 8:** Продавец проверяет прекурсор и отправляет его банку, который также проверяет, получена ли купюра x из x'. В случае окончательного подтверждения сделки банк записывает купюру как депонированную, сохраняет копию прекурсора и переводит деньги на счет продавца.

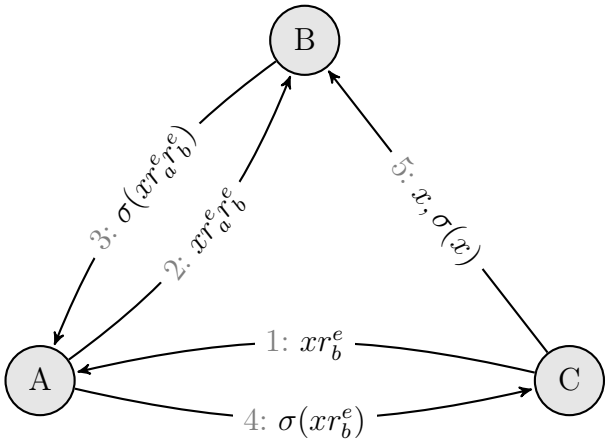
4. Реализуйте простое клиент-серверное приложение, позволяющее реализовывать электронные деньги согласно следующей схеме [обеспечение неотслеживаемости продавца]:



Здесь: A – банк, B – клиент, C – продавец, x – купюра, r – известное только продавцу случайное число, e – открытый ключ банка.

Шаг 1: Продавец производит цифровую купюру, затемняет и отправляет клиенту. **Шаг 2:** Клиент отправляет кюпуру банку. **Шаг 3:** Банк подписывает купюру, отправляет клиенту и снимает со счета клиента деньги. **Шаг 4:** Клиент пересылает подписанный «полуфабрикат» купюры продавцу. **Шаг 5:** Продавец с помощью открытого ключа банка проверяет, получена ли купюра из отправленной в банк, и после снятия своего затемняющего множителя отправляет купюру в банк (банк переводит деньги на счет продавца).

5. Реализуйте простое клиент-серверное приложение, позволяющее реализовывать электронные деньги согласно следующей схеме [неотслеживаемость клиента и продавца]:



Здесь: A – банк, B – клиент, C – продавец, x – купюра, r_b – известное только продавцу случайное число, r_a – известное только клиенту случайное число, e – открытый ключ банка.

Шаг 1: Продавец производит цифровую купюру, затемняет и отправляет клиенту. **Шаг 2:** Клиент дополнительно затемняет купюру и отправляет в банк. **Шаг 3:** Банк подписывает дважды затемненную купюру и отправляет клиенту. **Шаг 4:** Клиент снимает с подписанной купюры свой затемняющий множитель и отправляет продавцу. **Шаг 5:** Продавец снимает с подписанной купюры свой затемняющий множитель и отправляет банку.