Тематическое содержание дисциплины «Защита информации»[1]

1. Обзор доктрины информационной безопасности Российской Федерации.
2. Организационные меры защиты информации.
3. Обзор и практика применения федерального закона «О персональных данных» [изучается самостоятельно].
4. Элементарные шифры замены и перестановки. Многоалфавитный шифр Цезаря.
5. Совершенные шифры. Шифр Вернама.
6. Математические основы криптографии: конечные группы, кольца и поля.
7. Генерация псевдослучайных последовательностей.
8. Односторонние функции. Протокол Диффи-Хеллмана.
9. Функции хеширования, схемы применения.
10. Поточные шифры. Шифр RC4.
11. Блочные шифры, режимы применения. Шифры DES и Triple DES.
12. Шифр AES.
13. Шифры с открытым ключом и схемы их применения. Шифр RSA.
14. Криптография на эллиптических кривых. Схемы применения.
15. Технология Blockchain.

[1]Итоговая оценка определяется суммой баллов, набранных за семестр, включая ответ по билету (до 19 баллов): «не зачтено» – от 0 до 60 баллов; «зачтено» – от 61 до 100 баллов. Все работы (девять) подлежат защите. Каждая работа оценивается по шкале от 0 до 3 баллов, полученный результат умножается на коэффициент сложности и идет в общий зачет. Работы следует направлять на адрес: elenakhaa@yandex.ru. В условиях повторной промежуточной аттестации неоходимым условием получения положительной оценки за зачет является получение за ответ на зачете не менее 11 баллов. Пример задания билета: «Опишите режимы применения блочных шифров». Время подготовки к ответу – 15 минут.

Задачи на тему: «Группы, кольца и поля»[1] [актуальны только до 1 июня, коэффициент сложности равен 3]

1. Покажите, образует ли множество $\{-1, 1\}$ группу по умножению.
2. Покажите, образует ли множество $\{0\}$ группу по умножению.
3. Покажите, образует ли множество четных чисел группу относительно операции сложения.
4. Покажите, является ли множество отличных от нуля вещественных чисел группой по сложению.
5. Покажите, является ли множество отличных от нуля вещественных чисел группой по умножению.
6. Покажите, является ли множество положительных вещественных чисел группой по сложению.
7. Покажите, является ли множество положительных вещественных чисел группой по умножению.
8. Покажите, является ли множество неотрицательных вещественных чисел группой по сложению.
9. Покажите, является ли множество неотрицательных вещественных чисел группой по умножению.
10. Покажите, является ли множество целых положительных степеней двойки [2, 4, 8 и т.д.] группой по сложению.
11. Покажите, является ли множество целых положительных степеней двойки [2, 4, 8 и т.д.] группой по умножению.

[1] Итоговая оценка определяется суммой баллов, набранных за семестр, включая ответ по билету (до 19 баллов): «не зачтено» – от 0 до 60 баллов; «зачтено» – от 61 до 100 баллов. Все работы (девять) подлежат защите. Каждая работа оценивается по шкале от 0 до 3 баллов, полученный результат умножается на коэффициент сложности и идет в общий зачет. Работы следует направлять на адрес: elenakhaa@yandex.ru. В условиях повторной промежуточной аттестации неоходимым условием получения положительной оценки за зачет является получение за ответ на зачете не менее 11 баллов [условная «удовлетворительно»]. Пример задания билета: «Опишите режимы применения блочных шифров». Время подготовки к ответу – 15 минут.

12. Покажите, является ли множество всех степеней двойки [с целым положительным, целым отрицательным и нулевым показателем] группой по сложению.

13. Покажите, является ли множество всех степеней двойки [с целым положительным, целым отрицательным и нулевым показателем] группой по умножению.

14. Покажите, является ли множество направленных в одну и ту же сторону векторов в пространстве коммутативной группой относительно сложения векторов.

15. Покажите, является ли группой множество подстановок из трех элементов относительно композиции подстановок.

16. Покажите, является ли группой множество таких подстановок из четырех элементов, которые единицу переводят в единицу, относительно композиции подстановок.

17. Покажите, является ли множество линейных функций группой по операции линейной замены независимой переменной. Линейной называется функция вида $y = kx + b$, где a и b – вещественные числа, причем $a \neq 0$.

18. Покажите, является ли множество целых чисел группой по операции возведения в степень.

19. Покажите, является ли множество положительных вещественных чисел группой по операции возведения в степень.

20. Покажите, является ли множество четных чисел кольцом по операциям сложения и умножения.

21. Покажите, является ли множество чисел, кратных трем, кольцом по операциям сложения и умножения.

22. Постройте некоммутативное кольцо, состоящее из четырех элементов.

23. Найдите несколько целочисленных решений (x, y) уравнения $2x + 3y = 11$.

24. Покажите, является ли множество многочленов с целочисленными коэффициентами полем по операциям сложения и умножения многочленов.

25. Покажите, является ли множество многочленов с вещественными коэффициентами полем по операциям сложения и умножения многочленов.

Лабораторная работа 1: «Изучение свойств мультипликативной группы расширенного поля Галуа»[1]

Задание 1. Арифметика в $GF(2^m)$ [коэффициент сложности равен 1]

Многочленом степени k над полем $\mathbb{Z}_2$ называется выражение

$$a(x) = a_0 + a_1x + a_2x^2 + ... + a_kx^k,$$

где $a_i \in \mathbb{Z}_2$ и $a_k \neq 0$. Многочлен называется *неприводимым*, если его нельзя разложить на множители.

Операции сложения и умножения многочленов над полем $\mathbb{Z}_2$ отличаются от операций сложения и умножения многочленов над полем $\mathbb{R}$ только тем, что коэффициенты одночленов складываются и умножаются по правилам поля $\mathbb{Z}_2$.

Каждому многочлену над полем $\mathbb{Z}_2$ можно сопоставить класс вычетов по модулю неприводимого многочлена $p(x)$ степени m. Эти классы вычетов образуют конечное поле $GF(2^m)$ с определенными на нем операциями сложения и умножения многочленов: $\mathbb{Z}_2[x] \to \mathbb{Z}_2[x]/(p)$ по аналогии с $\mathbb{Z} \to \mathbb{Z}_p$. В поле $GF(2^m)$ входят всевозможные многочлены со степенями до $m-1$ и коэффициентами из $\mathbb{Z}_2$, т.е. мощность $GF(2^m)$ равна 2^m.

Сложение элементов поля $GF(2^m)$ сводится к сложению многочленов над полем $\mathbb{Z}_2$, а умножение элементов поля $GF(2^m)$ заключается в умножении многочленов над полем $\mathbb{Z}_2$ и взятию остатка от деления результата на многочлен $p(x)$.

Например, если поле $GF(2^m)$ образовано $p(x) = x^4 + x + 1$, то:

$$(x+1) + (x^2 + x) = x^2 + 1,$$

$$x^2 \cdot (x^2 + 1) = x^2 + x + 1.$$

Задание:

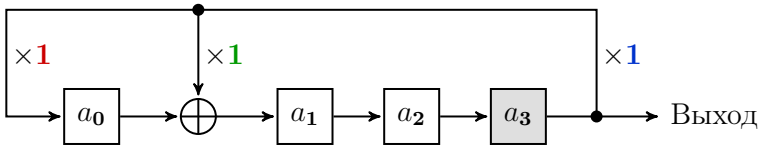
1. Пусть $p(x) = x^4 + x$, вычислите x^0, x^1, ..., x^{16}. Покажите, что 0 и всевозможные степени многочлена x образуют кольцо, но не поле.

2. Пусть $p(x) = x^4 + x + 1$, вычислите x^0, x^1, ..., x^{16}. Покажите, что 0 и всевозможные степени многочлена x образуют поле.

[1] Необходимо выполнить и защитить одно из предложенных заданий. Решение нужно сопроводить содержательным отчетом (в произвольной форме).

Задание 2. Статистические свойства генерируемой РСЛОС последовательности [коэффициент сложности равен 2]

1. Опишите выработку псевдослучайной последовательности битов на основе регистра сдвига с линейной обратной связью (РСЛОС) в конфигурации Галуа. Выпишите в виде двоичных векторов последовательные состояния РСЛОС, соответствующего образующему многочлену $p(x) = x^4 + x + 1 = 1 + 1 \cdot x^1 + 0 \cdot x^2 + 0 \cdot x^3 + 1 \cdot x^4$ [ненулевые коэффициенты определяют позиции отводов]:



2. Выведите расчетную формулу.

3. С помощью критерия χ^2 оцените качество любой генерируемой псевдослучайной последовательности, длина которой больше и кратна максимальной (см. `knuth_chi2.pdf`).

Справка. Комбинация из трех регистров сдвига применяется для генерации ключевой последовательности в A5 – поточном алгоритме шифрования, используемом для обеспечения конфиденциальности передаваемых данных между телефоном и базовой станцией в европейской системе мобильной цифровой связи GSM.

Задание 3. Программная реализация РСЛОС [коэффициент сложности равен 3]

Разработайте консольное приложение для генерации псевдослучайной последовательности битов на основе регистра сдвига с линейной обратной связью (РСЛОС) в конфигурации Галуа (в нашем случае над полем $\mathbb{Z}_2$). Начальное значение сдвигового регистра и его образующий многочлен должны быть ограничены длиной машинного слова и задаваться пользователем. Варианты «легковесных» неприводимых многочленов приведены в файле `HPL-98-135.pdf` [например, запись $(4, 1)$ следует трактовать как $x^4 + x + 1$].

Примечание. Зная длину m регистра сдвига с линейной обратной связью и $2m$ последовательных битов, вышедших из него, можно вычислить весь генерируемый поток.

Лабораторная работа 2: «Применение криптографических функций хеширования»[1]

О деталях реализации и средствах разработки

1. OpenSSL – криптографическая библиотека с открытым исходным кодом (написана на языке C). В приложение Git входит утилита `openssl.exe`: "C:\Program Files\Git\usr\bin\openssl.exe"(удобно вызывать в консоли через предварительно созданную переменную среды `openssl`).

2. Вычисление хешкода файла (с выводом в консоль и в файл соответственно):

```
openssl dgst -sha1 filename.in
```

```
openssl dgst -sha1 filename.in > dgst_filename.out
```

   Справка по команде `dgst` (для других команд – аналогично):

```
openssl help dgst
```

3. Генерация секретного ключа – всевдослучайного числа (здесь длина ключа – 256 бит):

```
openssl rand -hex 32 > key.bin
```

   Или:

```
openssl rand -out key.bin 32
```

   Вычисление HMAC[2] файла:

[1] Необходимо выполнить и защитить одно из предложенных заданий. Решение нужно сопроводить содержательным отчетом (в произвольной форме).

[2] HMAC (англ. Hash-based Message Authentication Code) – *имитовставка* на основе хеширования. Добавляется к сообщению для его аутентификации. Длина имитовставки равна длине хешкода базовой функции хеширования.

```
openssl mac -digest SHA-256 -macopt hexkey:<($key) -in filename.in
↪  HMAC < key.bin
```

4. Генерация файла с параметрами для последующей генерации ключа по протоколу Диффи-Хеллмана (для ключа в 2048 бита):

```
openssl dhparam -out params.pem 2048
```

Просмотр файла с параметрами:

```
openssl dhparam -in params.pem -text -noout
```

Генерация ключа в формате PEM (в кодировке Base64):

```
openssl genpkey -paramfile params.pem -out privatekey.pem
```

Просмотр ключа:

```
openssl pkey -in privatekey.pem -noout -text
```

Конвертация ключа в формат DER (бинарный):

```
openssl x509 -outform der -in privatekey.pem -out privatekey.der
```

5. В качестве HEX-редактора удобно использовать Notepad++ (требуется установить соответствующий плагин).

6. Вызов внешней утилиты `openssl.exe` из программы, написанной на языке Java[3]:

```
ProcessBuilder process = new ProcessBuilder();
process.command("C:\\...\\openssl.exe","arg1","arg2", ...);
process.start();
```

[3]В этой и последующих работах в качестве криптографического инструментария допустимо использование только утилиты `openssl.exe`.

7. В Java определены два типа потоков ввода/ввывода – байтовый и символьный, для ввода/вывода бинарного кода и символов (в кодировке Unicode) соответственно. На самом низком уровне ввод/вывод данных является байтовым. Чтобы при чтении/записи не утратить информацию, не имеющую символьного представления, следует использовать байтовые потоки.

Задание 1. Применение HMAC в сетевых протолах [коэффициент сложности равен 1]

HMAC порождает значение той же длины, что и базовая функция хеширования, и вычисляется по правилу[4]:

$$\text{HMAC}(\text{message}, K) = H(\text{opad} \oplus K' \,||\, H(\text{ipad} \oplus K' \,||\, \text{message})),$$

где: message – сообщение, подлежащее аутентификации; H – функция хеширования (например, SHA3-256); K – секретный ключ; K' – ключ длины, равной размеру блока, сформированный на основе K и зависящий от длины внутреннего блока функции хеширования, B (как правило, не совпадает с длиной генерируемого хеша); ipad – внутреннее дополнение, состоящее из байта 0x36, повторенного B раз; opad – внешнее дополнение, состоящее из байта 0x5C, повторенного B раз; $||$ – символ конкатенации.

K' формируется из K следующим образом: если длина K меньше или равна B, то K' совпадает с K, дополненным байтами 0x00 до длины B; если длина K больше B, то K' совпадает с $H(K)$, дополненным байтами 0x00 до длины B.

Общий порядок применения MAC: Алиса отправляет Бобу сообщение message вместе с $T = \text{MAC}(\text{message}, K)$; Боб получает message и T, вычисляет $\text{MAC}(\text{message}, K)$ и проверяет, совпадают ли вычисленное значение с полученным; если значения имитовставок равны, Боб удостоверяется в том, что сообщение не было подделано Мэллори.

Задание: опишите общие схемы защищенной передачи данных с помощью сетевых протоколов TLS, SSH и IPsec [алгоритмы симметричного и ассиметричного шифрования раскрывать не нужно – принципиально]; объясните назначение и механику применения HMAC в них, а также порядок выработки ключей.

[4] По сути: $\text{hmac}(\text{message}, K) = \text{hash}(\text{hash}(\text{message}) + K)$.

Задание 2. Авторизация пользователей по хеш-кодам [коэффициент сложности равен 2]

Разработайте простое веб-приложение, предусматривающее только регистрацию пользователей и их авторизацию по хеш-кодам паролей. Хеш-коды паролей должны храниться в базе данных на сервере [пароли храниться не должны]. Перед вычислением хеш-кода пароль должен быть дополнен длинной (не менее 128 бит) «солью» – в начале или в конце, единообразно для всех паролей [одинаковые пароли будут давать разные хеш-коды]. Для каждого пароля каждого пользователя соль долна быть получена криптографически стойким генератором псевдослучайных чисел (при смене пароля следует поменять и соль)[5]. Соль также должна храниться в базе данных на сервере (рядом с хеш-кодом). При авторизации хеш-код дополненного солью пароля должен вычисляться на стороне сервера и сравниваться с хранимым значением хеш-кода. При совпадении значений авторизация пользователя считается успешной.

Задание 3. Аутентификация сообщений по имитовставкам [коэффициент сложности равен 3]

Разработайте простое консольное клиент-серверное приложение, позволяющее клиенту и серверу обмениваться сообщениями, дополненными имитовставками на основе хеширования. Принимающая сторона должна по имитовставке проверить сообщение на подлинность и на целостность (аутентифицировать). Обмен ключами должен осуществляться по базовой схеме протокола Диффи-Хеллмана (на основе $\mathbb{Z}_p$, где p – простое число).

[5] Для справки: `java.security.SecureRandom`.

Лабораторная работа 3: «Применение блочных шифров – 1»[1, 2]

О деталях реализации и средствах разработки

1. Генерация ключа для применения шифра 3DES [с тремя различными ключами][3]:

   ```
   openssl rand -hex 24 > key.bin
   ```

   Преобразование двоичного файла (ключа в нашем случае) в формат BASE64:

   ```
   openssl base64 < key.bin > key.base64
   ```

   Обратное преобразование (декодирование):

   ```
   openssl base64 -d < key.base64 > key.bin
   ```

   Формирование 256-битного ключа из парольной фразы (здесь предварительно генерируется 128-битная соль):

   ```
   openssl rand -hex 16 > salt.bin
   openssl kdf -keylen 32 -kdfopt pass:password -kdfopt
   ↪  hexsalt:<($salt) -kdfopt n:65536 -kdfopt r:8 -kdfopt p:1 SCRYPT
   ↪  < salt.bin
   ```

2. Шифрование небольшого документа посредстом шифра 3DES (блоки шифруются независимо друг от друга, режим ECB):

   ```
   openssl enc -des-ede3-ecb -in plaintext.in -out ciphertext.bin
   ↪  -pass file:key.bin
   ```

[1] Необходимо выполнить и защитить одно из предложенных заданий. Решение нужно сопроводить содержательным отчетом (в произвольной форме).

[2] При необходимости использовать криптографические средства защиты информации следует ограничиться вызовом утилиты `openssl`.

[3] Triple DES, или 3DES, – шифр, сводящийся к троекратному применению шифра DES [устарел, но поддерживается для возможности работы со старыми серверами]. Варианты ключей: k_1, k_1 и k_3 различны; $k_1 = k_3$, а k_1 и k_2 различны; $k_1 = k_2 = k_3$. Российским аналогом являетя блочный шифр ГОСТ89, или Магма, – есть в OpenSSL.

Расшифрование документа, зашифрованного посредством шифра 3DES:

```
openssl enc -d -des-ede3-ecb -in ciphertext.bin -out plaintext.out
↪  -pass file:key.bin
```

По умолчанию перед шифрованием добавляется соль [для противодействия обращения шифртекста с помощью радужных таблиц, уместно при шифровании хранимых паролей], с ключом `-nosalt` соль добавлена не будет.

Использование ключа `-base64` приведет к представлению зашифрованных (двоичных) данных в формате ASCII. Перед расшифрованием данных потребуется их декодирование (с тем же ключом).

3. Шифрование большого документа посредстом шифра 3DES (блоки шифруются «внахлест», режим CBC):

```
openssl enc -des-ede3-cbc -in plaintext.in -out ciphertext.bin
↪  -pass file:key.bin
```

Или (режим шифрования CBC):

```
openssl enc -des3 -in plaintext.in -out ciphertext.bin -pass
↪  file:key.bin
```

Задание 1. Применение симметричных шифров в сетевых протоколах [коэффициент сложности равен 1]

Опишите общие схемы защищенной передачи данных с помощью сетевых протоколов TLS, SSH и IPsec [алгоритмы асимметричного шифрования раскрывать не нужно – принципиально]; объясните назначение и механику применения симметричных шифров в них, а также порядок выработки ключей.

Задание 2. Алгоритм шифра DES [коэффициент сложности равен 2]

Распишите (на примере) и поясните процедуру применения шифра S-DES, или Simplified DES [файл `simplified_DES.pdf`], для шифрования одного блока открытого сообщения и расшифрования одного блока

шифртекста. Входные данные: открытое сообщение – 01110010, ключ – 1010000010.

Задание 3. Обмен зашифрованными сообщениями с имитовставками [коэффициент сложности равен 3]

Разработайте простое консольное клиент-серверное приложение, позволяющее клиенту и серверу обмениваться зашифрованными сообщениями с возможностью проверять аутентичность сообщений еще до их расшифрования: шифруется открытое сообщение – для шифртекста вычисляется MAC – шифртекст отправляется вместе с MAC. В качестве шифра следует выбрать Triple DES, в качестве алгоритма вычисления имитовставки – HMAC. Обмен ключами для вычисления имитовставки должен осуществляться по базовой схеме протокола Диффи-Хеллмана.

Лабораторная работа 4: «Применение блочных шифров – 2»[1, 2]

О деталях реализации и средствах разработки

1. Генерация случайного числа для использования в качестве ключа (длина ключа – 128 бит):

```
openssl rand -hex 16 > key.bin
```

2. Шифрование файла с помощью шифра AES (длина блока – 256 бит, режим шифрования – CBC):

```
openssl enc -aes-256-cbc -in message.in -out message.enc -pass
↪  file:key.bin
```

   По умолчанию утилита `openssl` извлекает инициализационный вектор из предоставленного пароля. Чтобы указать отдельно ключ и инициализационный вектор, нужно вместо `-pass` использовать `-K` и `-iv` соотвественно.

3. Расшифрование файла, зашифрованного с помощью шифра AES:

```
openssl enc -d -aes-256-cbc -in message.enc -out message.dec -pass
↪  file:key.bin
```

Задание 1. Место полей Галуа в шифре AES [коэффициент сложности равен 1]

- **SubBytes**, $GF(2^8)$: выполните процедуру в части вычисления мультипликативного обратного для заданного байтом $\{44\}$ многочлена.
- **MixColumns**, $GF(2^4)$: произведите процедуру над заданным байтами $\{d4\ bf\ 5d\ 30\}$ многочленом.

[1] Необходимо выполнить и защитить одно из предложенных заданий. Решение нужно сопроводить содержательным отчетом (в произвольной форме).

[2] При необходимости использовать криптографические средства защиты информации следует ограничиться вызовом утилиты `openssl`.

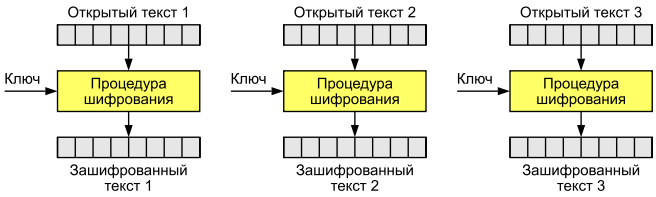
Задание 2. Алгоритм шифра AES [коэффициент сложности равен 2]

Распишите (на примере) и поясните процедуру применения шифра S-AES, или Simplified AES [файл `simplified_AES.pdf`], для шифрования одного блока открытого сообщения и расшифрования одного блока шифртекста. Входные данные: открытое сообщение – 0100 1010 1111 0101, ключ – 1101 0111 0010 1000.

Задание 3. Режимы применения блочных шифров [коэффициент сложности равен 3]

Для блочных шифров определены универсальные режимы работы. Основные из них:

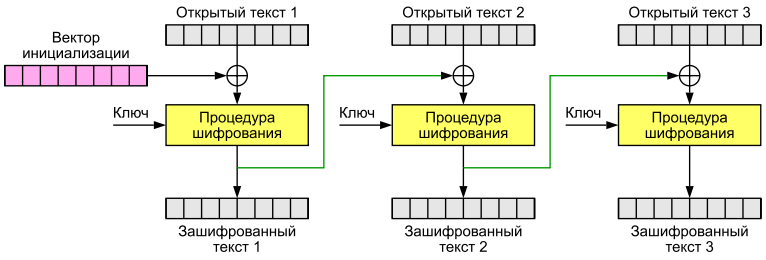
1. **ECB** (Electronic Codebook) – режим простой замены. Шифрование:



   Блоки открытого текста шифруются независимо друг от друга на одном и том же ключе. Режим применим для шифрования небольших объемов данных (например, ключей шифрования)[3]. Расшифрование:

$$\boldsymbol{m_i} = \boldsymbol{D_k}(c_i).$$

2. **CBC** (Cipher Block Chaining) – режим простой замены с зацеплением. Шифрование:



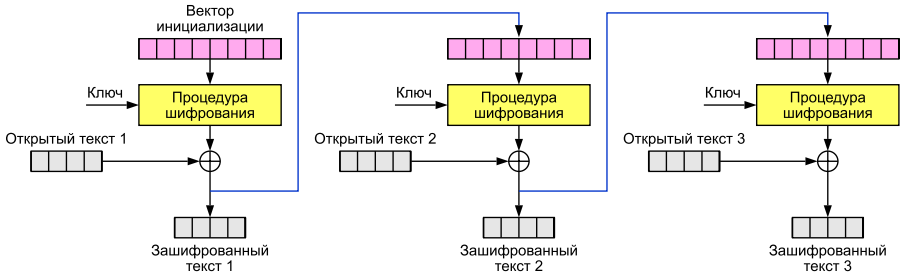
[3]Одинаковые блоки открытого текста представляются одинаковыми блоками в шифртексте.

На вход алгоритму шифрования поступает результат сложения по модулю два текущего блока открытого текста с предшествующим блоком шифртекста. Режим общего назначения, используется с целью обеспечения конфиденциальности. Расшифрование:

$$c_i = E_k(c_{i-1} \oplus m_i),$$
$$\boldsymbol{m_i} = \boldsymbol{D_k}(c_i) = D_k(E_k(c_{i-1} \oplus m_i)) = c_{i-1} \oplus m_i =$$
$$= c_{i-1} \oplus D_k(c_i) = c_{i-1} \oplus c_{i-1} \oplus m_i = m_i.$$

Для получения первого блока шифртекста требуется секретный *инициализационный вектор*, который должен быть известен и отправителю, и получателю сообщения [при передаче может шифроваться блочным шифром в режиме ECB][4].

3. **CFB** (Cipher Feedback) – режим гаммирования с обратной связью по шифртексту. Шифрование:



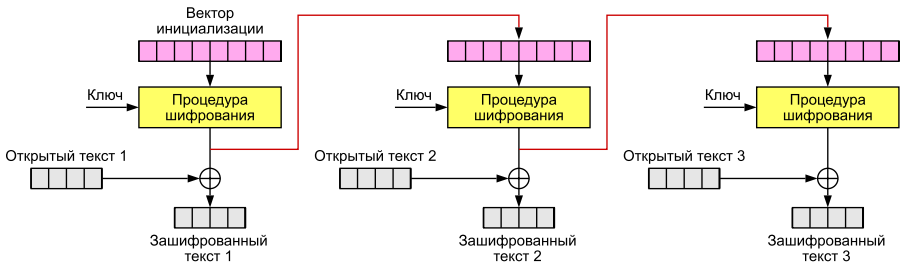
Режим использует блочный шифр в качестве поточного. Единицей передачи информации является один байт [как правило]. Сцепление блоков осуществляется по схеме CBC. Алгоритм блочного шифрования используется для выработки псевдослучайной последовательности. Сначала инициализационный вектор (64 бита) записывается в регистр сдвига и подается на вход алгоритму блочного шифрования. Левая часть (8 бит) полученного результа используется в качестве ключа для поточного шифрования поступившего на вход открытого сообщения (8 бит) – сложением по модулю

[4]Если злоумышленник может подменить инициализационный вектор, то может инвертировать избранные биты в первом блоке открытого сообщения: $c_1 = E_k(\mathrm{iv} \oplus m_1)$, $m_1' = \mathrm{iv}' \oplus D_k(c_1)$, здесь m_1 – отправляемое сообщение, iv' – навязанный (определенный) инициализационный вектор, а m_1' – получаемое (измененное) при расшифровании сообщение.

два. Для шифрования следующего блока открытого сообщения регистр сдвига претерпевает изменения (сдвиг влево на 8 бит и замена справа 8 бит): $\mathrm{iv}_2 = \mathrm{right56}(\mathrm{iv}_1) \,||\, c_1$. Расшифрование первого блока (остальные – аналогично):

$$\begin{gathered}\mathrm{iv}_1 = \mathrm{iv},\\ c_1 = m_1 \oplus \mathrm{left8}(E_k(\mathrm{iv}_1)),\\ \boldsymbol{m_1} = c_1 \oplus \mathrm{left8}(\boldsymbol{E_k}(\mathrm{iv}_1)).\end{gathered}$$

4. **OFB** (Output Feedback) – режим гаммирования с обратной связью по выходу. Шифрование:



Режим повторяет режим CFB за тем исключением, что в регистр сдвига подается значение, полученное на выходе алгоритма блочного шифрования. Используется в качестве поточного при передаче информации по ненадежным каналам, с помехами (например, по спутниковой связи)[5].

Задание. Конвертируйте исходное изображение `tux.png` в формат BMP и разработайте консольное приложение для его шифрования с помощью шифра AES. В учебных целях заголовочную часть файла зашифровывать не нужно. Режимы шифрования: ECB, CBC, CFB и OFB [нужно получить четыре варианта зашифрованного изображения]. Сравните результаты шифрования и скорости выполнения алгоритмов.

[5]Преимущество перед CFB: влияние возможных искажений битов при передаче данных не распространяется на следующие порции данных. Например, если искаженные биты появились при передаче c_i, это повлияет на восстановление из c_i значения m_i, но остальная часть открытого сообщения повреждена не будет. Недостаток: подмена одного бита открытого сообщения повлечет изменение в соответствующем бите восстановленного открытого сообщения [возможны контролируемые изменения в получаемом сообщении].

Лабораторная работа 5: «Применение шифров с открытым ключом – 1»[1, 2]

О деталях реализации и средствах разработки

1. Генерация закрытого/секретного ключа [пары ключей]:

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

   По умолчанию ключ содержит 4096 бит, произвольный размер устанавливается через `pkeyopt`. PEM-ключ можно просмотреть в любом plain-редакторе.

2. Извлечение открытого/публичного ключа из закрытого:

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

3. Вывод информации о структуре закрытого ключа [значение модуля n, простые множители p и q, экспоненты e и d] и отдельно значения модуля:

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

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

4. Шифрование файла:

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

5. Расшифрование файла:

[1] Необходимо выполнить и защитить одно из предложенных заданий. Решение нужно сопроводить содержательным отчетом (в произвольной форме).

[2] При необходимости использовать криптографические средства защиты информации следует ограничиться вызовом утилиты `openssl`.

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

6. Вычисление цифровой подписи файла:

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

   Зашифровывается хешкод файла (команды можно разделить).

7. Верификация цифровой подписи файла:

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

   В случае неудачной верификации: «Verification Failure». Иначе: «Signature Verification Failure».

Задание 1. Нахождение мультипликативного обратного элемента в кольце $\mathbb{Z}_n$ [коэффициент сложности равен 1]

1. Реализуйте расширенный алгоритм Евклида.
2. На основе расширенного алгоритма Евклида реализуйте алгоритм вычисления обратного по умножению для элемента из кольца классов вычетов. Нужно учесть, что в кольце не для любого ненулевого элемента существует мультипликативный обратный. Основа решения следующая. Для взаимно простых r и n справедливо, что НОД$(r, n) = 1$:

$$rx + ny = 1 \Rightarrow rx \equiv 1 \pmod{n}.$$

Задание 2. Мультипликативность шифра RSA [коэффициент сложности равен 2]

Докажите, что криптосистема RSA мультипликативна, т.е.[3]:

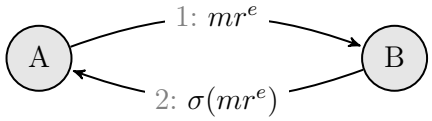
$$P_A(M_1) \cdot P_A(M_2) \equiv P_A(M_1 \cdot M_2) \pmod{n}$$

[3]По-другому: $(m_1^e \mod n) \cdot (m_2^e \mod n) \equiv (m_1 \cdot m_2)^e \pmod{n}$.

для любых $M_1, M_2 \in \mathbb{Z}_n$. Пусть злоумышленник имеет возможность быстро расшифровать 1% всех сообщений из $\mathbb{Z}_n$. Используя свойство мультипликативности системы RSA, объясните, как с высокой вероятностью злоумышленник может достаточно быстро расшифровать любое сообщение[4].

Задание 3. Слепая цифровая подпись [коэффициент сложности равен 3]

Принципиальная схема получения слепой цифровой подписи[5] [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:

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

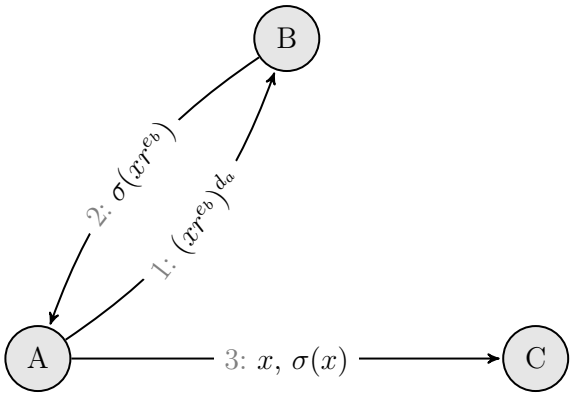
где (d, n) – закрытый ключ участника B. Чтобы получить подпись участника B документа m, участник A должен умножить $\sigma(mr^e)$ на число, обратное r.

Задание. Реализуйте простое консольное клиент-серверное приложение, позволяющее аккумулировать короткие анонимные сообщения [или простую систему анонимного электронного голосования] согласно следующей схеме:

---

[4]Задача со звёздочкой из книги с ослом: Т. Кормен, Ч. Лейзерсон, Р. Ривест, *«Алгоритмы: построение и анализ»*.

[5]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.



Здесь: A – пользователь [или избиратель], B – регистратор участников, C – сборщик [или счетчик] голосов, x – сообщение [или голос], r – известное только участнику A случайное число, (e_b, n) – открытый ключ регистратора, (d_a, n) – закрытый ключ участника. Пренебрегите реализацией правильных механизмов распределения, хранения и сертификации ключей: открытый ключ A известен B, открытый ключ B известен A[6].

[6]Сообщение до шифрвания на d_a может дополняться вспомогательной информацией – именем обращающегося пользователя, типом операции и т.д.

Лабораторная работа 6: «Применение шифров с открытым ключом – 2»[1, 2]

О деталях реализации и средствах разработки

1. Реализация алгоритма быстрого возведения в степень в кольце вычетов $\mathbb{Z}_n$, $a^b \bmod n$:

```
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;
}
```

Задание 1. Возведение в степень в кольце вычетов [коэффициент сложности равен 1]

На основе китайской теоремы об остатках реализуйте оптимизированный алгоритм быстрого возведения в степень для случая кольца классов вычетов:

$$M^e \bmod (p \cdot q),$$

где p и q – простые числа.

Задание 2. Электронные деньги [коэффициент сложности равен 2]

Изучите главу 9 книги: Иванов М.А., Чугунков И.В., «Криптографические методы защиты информации в компьютерных системах и сетях»[3].

[1] Необходимо выполнить и защитить одно из предложенных заданий. Решение нужно сопроводить содержательным отчетом (в произвольной форме).

[2] При необходимости использовать криптографические средства защиты информации следует ограничиться вызовом утилиты `openssl`.

[3] Или главу 11 из: Деднев М.А., Дыльнов Д.В., Иванов М.А., «Защита информации в банковском деле и электронном бизнесе».

Опишите и объясните принципиальные схемы применения цифровой подписи для обеспечения (B – банк):

- неотслеживаемости покупателя (A);
- неотслеживаемости продавца (C);
- неотслеживаемости покупателя (A) и продавца (C).

Задание 3. Сквозное шифрование [4] [коэффициент сложности равен 3]

При сквозном шифровании зашифровывается только передаваемое по сети сообщение, шифртекст дополняется необходимой для маршрутизации сообщения служебной информацией. Само сообщение остается зашифрованным на всем пути от отправителя к получателю (не расшифровывается на сервере), служебная же информация передается в незашифрованном виде[5]. Сквозное шифрование никак не скрывает сам факт передачи сообщения [известны дата, время, отправитель, получатель, сеть; периодичность контактов и структура данных информативны для криптоаналитика]. При сквозном шифровании также невозможно хранить историю переписки на сервере или использовать автоматическую модерацию беседы.

Принципиальная схема обмена данными на основе сквозного шифрования:

1. При создании сеанса связи получатель генерирует публичный и закрытый ключи, публичный передает на сервер.
2. Отправитель запрашивает у сервера публичный ключ получателя, шифрует на нем сообщение и отправляет его на сервер.
3. Сервер направляет полученное от отправителя зашифрованное сообщение получателю.
4. Получатель расшифровывает сообщение на своем закрытом ключе.

[4] *Сквозное шифрование* (от англ. *End-to-End Encryption*, *E2EE*) – способ защищенного обмена данными, при котором шифрование и расшифрование информации выполняется только на конечных устройствах пользователей.

[5] При канальном шифровании зашифровываются все – и открытое сообщение, и данные о маршрутизации. Для передачи данных от отправителя к получателю сервер-посредник должен их сначала их расшифровать, а затем заново зашифровать.

Задание. Реализуйте простое консольное клиент-серверное приложение для обмена сообщениями между двумя клиентами по следующей схеме. Пусть пользователь A отправляет сообщение пользователю B (причем пользователи заранее сообщили серверу свои открытые ключи):

1. Пользователь A генерирует одноразовый AES-ключ и зашифровывает на нем свое сообщение.

2. Пользователь A запрашивает у сервера открытый RSA-ключ пользователя B и зашифровывает на нем одноразовый ключ.

3. Пользователь A подписывает на своем закрытом ключе комбинацию из зашифрованных одноразового ключа и сообщения, после чего отправляет на сервер (при вычислении подписи комбинация хешируется).

4. Сервер получает сообщение от пользователя A и перенаправляет его пользователю B.

5. Пользователь B получает сообщение, сверяет подпись, расшифровывает одноразовый ключ, затем расшифровывает само сообщение.

При отправке сообщения от пользователя B к пользователю A схема аналогичная.

Лабораторная работа 7: «Применение шифров с открытым ключом – 3»[1, 2]

О деталях реализации и средствах разработки

1. Обмен ключами с использованием эллиптических кривых:

1.0. Для обмена ключами по схеме Диффи-Хеллмана сначала нужно задать эллиптическую группу точек $E_p(a, b)$, затем – выбрать генерирующую точку $G(x_1, y_1)$. Важно, чтобы наименьшее значение k, при котором $kG = \mathcal{O}$, оказалось достаточно большим простым числом. Значения p, a, b и G считаются общеизвестными. Процедура обмена ключами между Алисой и Бобом: 1) Алиса выбирает секретное целое число $k_A < p$ и генерирует точку $P_A = k_A G$; 2) Боб выбирает секретное целое число $k_B < p$, затем генерирует точку $P_B = k_B G$; 3) Алиса и Боб обмениваются вычисленными P_A и P_B; 4) Алиса и Боб независимо друг от друга вычисляют секретный ключ K: $k_A P_B = k_A(k_B G) = k_B(k_A G) = k_B P_A$. Злоумышленнику вычислить k по известным G и kG вычислительно сложно. Для использования в качестве ключа шифрования используют только одну координату полученной точки.

1.1. Вывод списка эллиптических кривых, используемых `openssl`:

```
openssl ecparam -list_curves
```

1.2. Генерация файла с параметрами для последующей генерации ключа по протоколу Диффи-Хеллмана на эллиптических кривых (название кривой – любое из списка):

```
openssl ecparam -name prime256v1 -genkey -noout -out
↪  params.pem
```

1.3. Просмотр файла с параметрами:

[1] Необходимо выполнить и защитить одно из предложенных заданий. Решение нужно сопроводить содержательным отчетом (в произвольной форме).

[2] При необходимости использовать криптографические средства защиты информации следует ограничиться вызовом утилиты `openssl`.

```
openssl ec -in params.pem -text -noout
```

1.4. Генерация закрытого ключа:

```
openssl ecparam -in params.pem -genkey -noout -out
↪  privatekey.pem
```

1.5. Этапы выработки ключа по протоколу Диффи-Хеллмана:

```
openssl genpkey -out alice.pem -algorithm EC -pkeyopt
↪  ec_paramgen_curve:P-256 -pkeyopt ec_param_enc:named_curve

openssl pkey -pubout -in alice.pem -out alice.pub

openssl genpkey -out bob.pem -algorithm EC -pkeyopt
↪  ec_paramgen_curve:P-256 -pkeyopt ec_param_enc:named_curve

openssl pkey -pubout -in bob.pem -out bob.pub

openssl pkeyutl -derive -out alicebob.key -inkey alice.pem
↪  -peerkey bob.pub

openssl pkeyutl -derive -out bobalice.key -inkey bob.pem
↪  -peerkey alice.pub
```

2. Шифрование с использованием эллиптических кривых:

2.0. Пусть открытое сообщение m передается в виде точки P_m: в общем случае невозможно закодировать сообщение одной координатой точки, т.к. такие точки могут отсутствовать в заданной эллиптической группе $E_p(a, b)$. Схема передачи зашифрованного сообщения от Алисы к Бобу: 1) Алиса выбирает секретное целое число $k_A < p$ и генерирует открытый ключ $P_A = k_A G$; 2) Боб выбирает секретное целое число $k_B < p$ и генерирует открытый ключ $P_B = k_B G$; 3) Алиса и Боб обмениваются вычисленными P_A и P_B; 4) Алиса выбирает случайное положительное целое число k, зашифровывает сообщение P_m и отправляет Бобу пару точек $C_m = \{kG, P_m + kP_B\}$; 5) Боб расшифровывает сообщение C_m, умножая первую точку на P_B и вычитая результат из второй точки: $P_m + kP_B - k_B(kG) = = P_m + k(k_B G) - k_B(kG) = P_m$. Алиса замаскировала сообщение P_m с помощью добавления к нему kP_B. Значение k знает

только Алиса, снять «маску» kP_B по известному P_B вычислительно сложно. Бобу известно значение k_B, для него Алиса включает в сообщение «подсказку». Злоумышленнику же вычислить k по известным G и kG крайне сложно.

2.1. Генерация пары EC-ключей с помощью утилиты `openssl` (максимальной длины, 570 бит):

```
openssl genpkey -algorithm EC -pkeyopt
↪  ec_paramgen_curve:secp521r1 -out keypair.pem
```

2.2. Просмотр файла с парой ключей (для подписания используют закрытый ключ, а для проверки подписи – открытый):

```
openssl pkey -in keypair.pem -noout -text
```

2.3. Извлечение открытого ключа (извлечь закрытый ключ можно только программно):

```
openssl pkey -in keypair.pem -pubout -out public_key.pem
```

2.4. Просмотр файла с открытым ключом:

```
openssl pkey -in public_key.pem -pubin -noout -text
```

2.5. Подписание файла (закрытый ключ подается в файле с парой ключей):

```
openssl pkeyutl -sign -digest sha3-512 -inkey keypair.pem -in
↪  file.txt -rawin -out file.signature
```

2.6. Проверка подписи:

```
openssl pkeyutl -verify -digest sha3-512 -inkey public_key.pem
↪  -in file.txt -rawin -sigfile file.signature
```

Задание 1. Свойства группы точек эллиптической кривой – 1 [коэффициент сложности равен 1]

1. Постройте эллиптическую кривую, удовлетворяющую уравнению

$$y^2 = x^3 - 12x + 20.$$

2. Вычислите группу точек эллиптической кривой $E(-12, 20)$ над полем $\mathbb{Z}_{37}$.

3. Изобразите точки, полученные в п. 2, и кривую, полученную в п. 1, в одной координатной плоскости.

4. Сложите в $E_{37}(-12, 20)$ точки $P(-2, -6)$ и $Q(1, 3)$, результат отметьте на полученной ранее диаграмме.

5. Удвойте в $E_{37}(-12, 20)$ точку $R(1, -3)$, результат отметьте на полученной ранее диаграмме.

Для проверки используйте desmos.

Задание 2. Свойства группы точек эллиптической кривой – 2 [коэффициент сложности равен 2]

Реализуйте алгоритм умножения натурального числа на точку, принадлежащую группе $E_p(a, b)$. Последовательно получите точки группы $E_{37}(-12, 20)$, начиная с произвольной точки группы: P, $2P$, $3P$, ... Подберите группообразующую точку.

Замечание. Умножение точки на число упрощается, если свести его к комбинации операций сложения и удвоения точек. Например [два сложения и два удвоения][3]:

$$7P = P + 6P = P + 2(3P) = P + 2(P + 2P).$$

Задание 3. Аутентификация по открытому ключу[4] [коэффициент сложности равен 3]

Разработайте простое ~~оконное~~ консольное приложение, которое разрешает авторизованным пользователям доступ к ресурсам сервера [например, чтение некоторых файлов]. Сервер хранит логин и открытый ключ пользователя, а также права доступа пользователя к ресурсам сервера. Схема аутентификации пользователя:

1) пользователь обращается к серверу (передает логин или свой открытый ключ), сервер отвечает пользователю коротким случайным сообщением;

[3] $E_p(a, b)$ – коммутативная по операции сложения точек группа.

[4] Аутентификация по открытому ключу используется в протоколе прикладного уровня SSH (от англ. *Secure Shell*), предназначенного для безопасного обмена информацией между двумя устройствами [над транспортным протоколом].

2) пользователь отправляет на сервер логин и зашифрованное на своем закрытом ключе полученное от сервера сообщение;

3) сервер расшифровывает на открытом ключе пользователя полученный шифртекст и сравнивает его с отправленным пользователю сообщением – в случае их совпадения разрешает пользователю доступ к ресурсам согласно имеющимся у него правам[5].

Для шифрования используйте методы криптографии на эллиптических кривых. Формат передачи запроса разработайте самостоятельно. Зашифровывать запрос клиента и ответ сервера не требуется [иначе дополнительно требуется выработка сеансового ключа для симметричного шифрования].

Задание 4. Выработка сеансового ключа [коэффициент сложности равен 3]

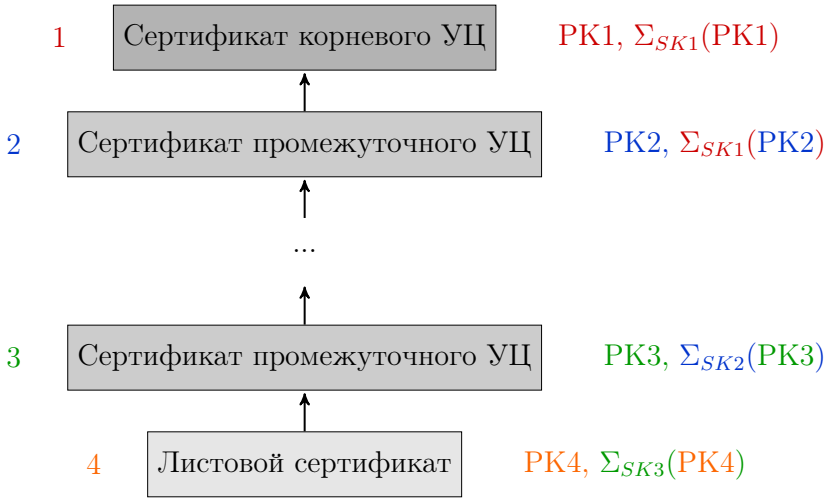
Разработайте простое консольное приложение, вырабатывающее между клиентом и сервером сеансовый ключ для последующего шифрования сообщений посредством симметричного шифра. Для генерации ключа используйте аналог протокола Диффи-Хеллмана, основанного на эллиптических кривых. Процедуру идентификации опуститите. Зашифровывать и передавать сообщения после выработки ключа не требуется.

[5]Возможна схема, когда сервер передает пользователю зашифрованное на открытом ключе пользователя случайное сообщение и ожидает от пользователя расшифрованный на закрытом ключе пользователя вариант этого сообщения.

Лабораторная работа 8: «Применение шифров с открытым ключом – 4»[1, 2]

О деталях реализации и средствах разработки

1. Цепочка сертификатов:



   Проверка идентичности с помощью сертификата состоит в проверке: 1) является ли лицо, предъявившее сертификат для идентификации, его владельцем; 2) является ли действительным предъявляемый сертификат[1]. Каждый сертификат, кроме корневого, выпущен и подписан сертификатом предыдущего удостоверяющего центра[2].

2. Создание самоподписываемого сертификата[3] (используется в качестве корневого УЦ и считается надежным):

[1] Необходимо выполнить и защитить одно из предложенных заданий. Решение нужно сопроводить содержательным отчетом (в произвольной форме).

[2] При необходимости использовать криптографические средства защиты информации следует ограничиться вызовом утилиты `openssl`.

[1] Аналогия: чтобы идентифицировать себя, человек предъявляет паспорт, но паспорт должен быть действительным.

[2] Если промежуточный сертификат скомпрометирован, то его можно отозвать, не отзывая корневой сертификат и остальные промежуточные сертификаты, подписанные тем же корневым.

[3] Для локальной сети целесообразно создание собственного/корпоративного центра сертификации открытых ключей.

2.1. Генерация пары ключей:

```
openssl genpkey -algorithm ED448 -out root_keypair.pem
```

2.2. Создание простого запроса на подписание сертификата и просмотр запроса:

```
openssl req -new -subj "/CN=Root CA" -addext
↪  "basicConstraints=critical,CA:TRUE" -key root_keypair.pem
↪  -out root_csr.pem
```

```
openssl req -in root_csr.pem -noout -text
```

2.3. Генерация сертификата (срок действия – 10 лет) и просмотр сертификата:

```
openssl x509 -req -in root_csr.pem -copy_extensions copyall
↪  -key root_keypair.pem -days 3650 -out root_cert.pem
```

```
openssl x509 -in root_cert.pem -noout -text
```

Сертификат самоподписываемый, поэтому поля `Issuer` и `Subject` одинаковы.

3. Создание несамоподписываемого сертификата (используется в качестве промежуточного УЦ и не считается надежным):

3.1. Генерация пары ключей:

```
openssl genpkey -algorithm ED448 -out intermediate_keypair.pem
```

3.2. Генерация запроса на подписание сертификата:

```
openssl req -new -subj "/CN=Intermediate CA" -addext
↪  "basicConstraints=critical,CA:TRUE" -key
↪  intermediate_keypair.pem -out intermediate_csr.pem
```

Поля `Issuer` и `Subject` отличаются.

3.3. Выпуск сертификата промежуточного УЦ, подписанного закрытым ключом корневого сертификата, и его просмотр:

```
openssl x509 -req -in intermediate_csr.pem -copy_extensions
↪  copyall -CA root_cert.pem -CAkey root_keypair.pem -days
↪  3650 -out intermediate_cert.pem
```

```
openssl x509 -in intermediate_cert.pem -noout -text
```

3.4. Выпуск листового сертификата (сообразно выпуску сертификата промежуточного УЦ), подписанного закрытым ключом сертификата промежуточного УЦ, и его вывод:

```
openssl genpkey -algorithm ED448 -out leaf_keypair.pem

openssl req -new -subj "/CN=Leaf" -addext
↪  "basicConstraints=critical,CA:FALSE" -key leaf_keypair.pem
↪  -out leaf_csr.pem

openssl x509 -req -in leaf_csr.pem -copy_extensions copyall
↪  -CA intermediate_cert.pem -CAkey intermediate_keypair.pem
↪  -days 3650 -out leaf_cert.pem
```

```
openssl x509 -in leaf_cert.pem -noout -text
```

Листовой сертификат не должен использоваться для выпуска других сертификатов, поэтому `CA:FALSE`.

4. Проверка сертификата (строится цепочка от проверяемого сертификата до надежного):

```
openssl verify -verbose -show_chain -trusted root_cert.pem
↪  -untrusted intermediate_cert.pem leaf_cert.pem
```

Флаг `-trusted` задает файл, содержащий один или несколько надежных сертификатов. Флаг `-untrusted` задает файл, содержащий один или несколько ненадежных сертификатов. Оба флага можно использовать несколько раз для задания нескольких файлов.

5. Проверка подлинности цифровой подписи.

Задание 1. Схемы анонимной цифровой подписи [коэффициент сложности равен 1]

Изучите и опишите порядок формирования и применения:

- *групповой* цифровой подписи;
- *кольцевой* цифровой подписи.

Задание 2. Выпуск и проверка сертификатов [коэффициент сложности равен 2]

В консоли с помощью команд утилиты `openssl.exe`:

1. Выпустите цепочку сертификатов.
2. Проверьте выпущенную цепочку сертификатов.
3. Проверьте один сертификат выпущенной цепочки.

Задание 3. Локальный удостоверяющий центр [коэффициент сложности равен 3]

Разработайте простое консольное клиент-серверное приложение, в котором сервер выступает в качестве удостоверяющего центра, а клиенты могут обмениваться подписанными документами (с возможностью проверки подписей). Для подписания документов используйте шифр RSA.

Иллюстрации[3]

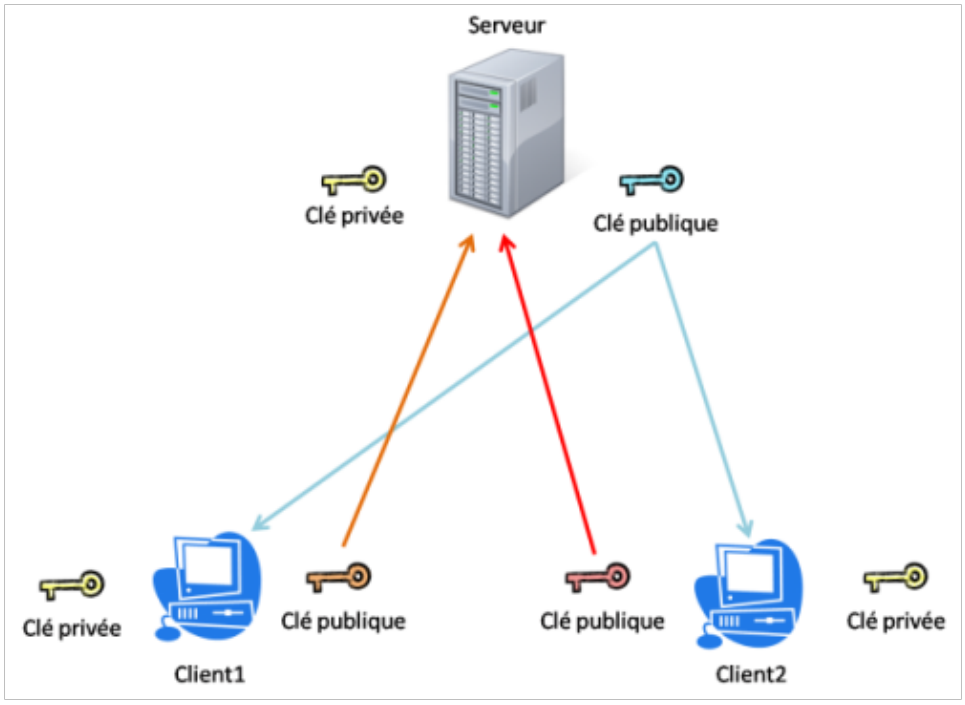


Рис. 1. Ключ, выбранный в качестве приватного, никогда никому не передается, в то время как ключ, выбранный в качестве публичного, может передаваться без ограничений. При установлении соединения (в рамках транзакции) различные участники обмениваются своими открытыми ключами друг с другом. Таким образом, Сервер владеет открытыми ключами Клиента 1 и Клиента 2, Клиент 1 и Клиент 2 владеют открытым ключом Сервера.

[3]Источник иллюстраций: remy-manu.no-ip.biz/C++/PDF/cryptage.pdf

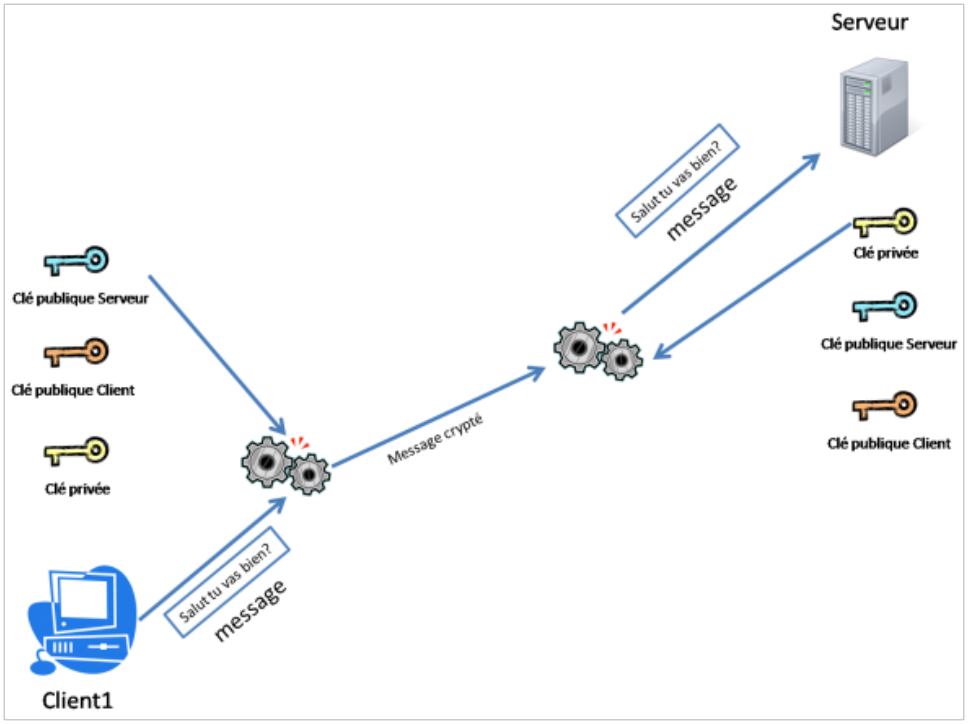


Рис. 2. Когда Клиент 1 вступает в диалог с Сервером (« Привет, ты в порядке?»), он шифрует сообщение с помощью открытого ключа Сервера. Только Сервер способен расшифровать сообщение, поскольку он единственный, у кого есть закрытый ключ.

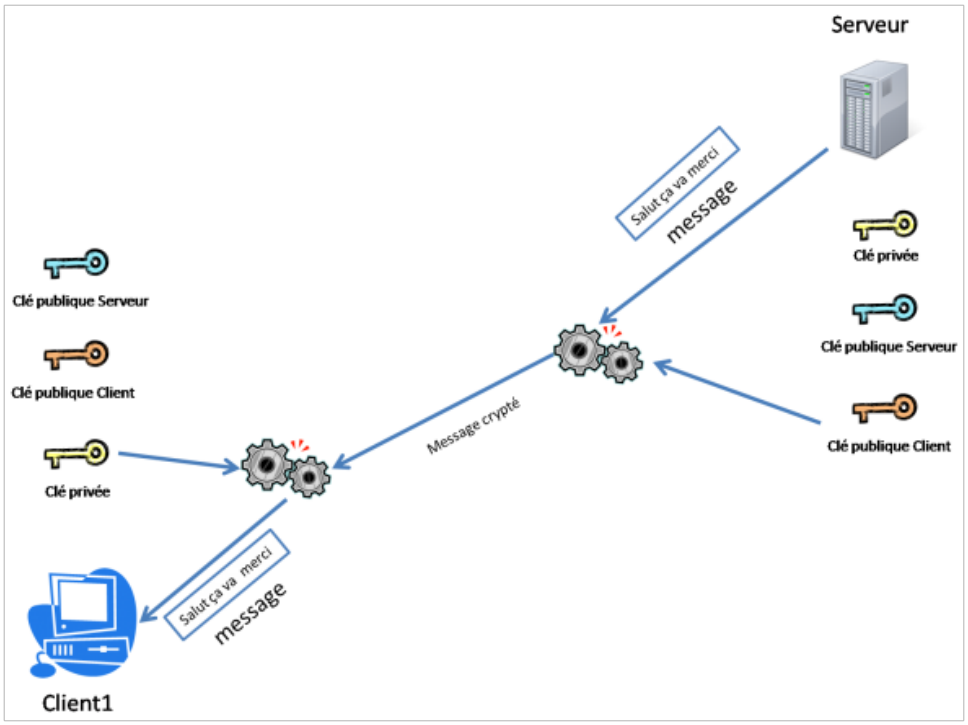


Рис. 3. Ответ Сервера Клиенту 1 происходит по аналогичной схеме.

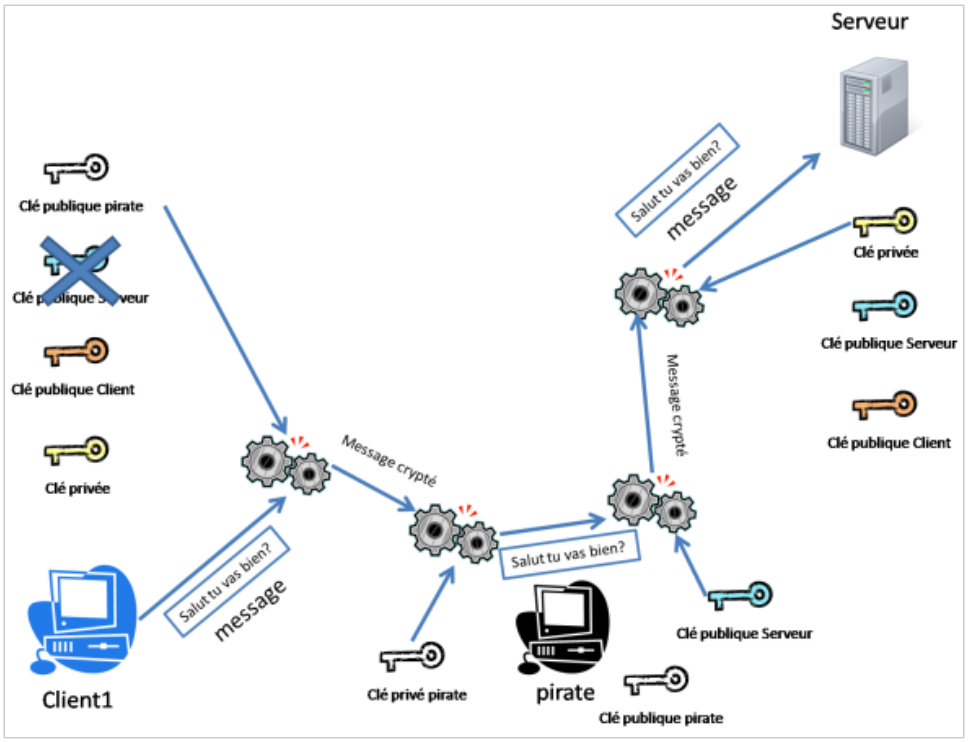


Рис. 4. Можно ли быть уверенным в использовании правильного открытого ключа для шифрования? Злоумышленник может встать между клиентом и сервером, передав клиенту свой открытый ключ. Так он сможет расшифровать все, что приходит от клиента. Для решения этой проблемы безопасности, существуют организации, называемые центрами сертификации, которые подписывают открытые ключи. Эта подпись позволяет подтвердить происхождение открытого ключа. Подписанный открытый ключ называется сертификатом.

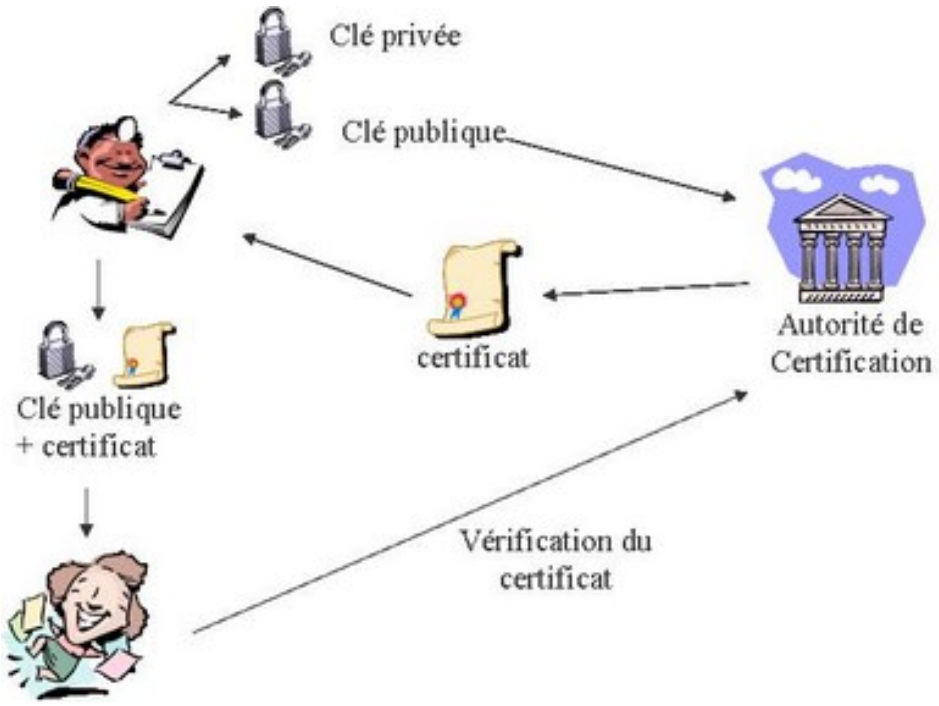


Рис. 5. Существуют общедоступные центры сертификации, подписывающие сертификаты своим закрытым ключом. Преимущество: большинство операционных систем имеют открытый ключ этих центров. Недостатком является то, что каждый выданный сертификат является платным и довольно дорогим. Для своей внутренней сети целесообразно создать собственный центр сертификации – достигаются экономия средств и независимость от сторонней компании в проверке сертификатов. Но: внешние клиенты не будут доверять таким сертификатам из-за отсутствия открытого ключа в операционной системе. В рамках внутренней сети это не проблема (блокировок): достаточно просто передать открытый ключ центра сертификации клиентам (хороший подход для обычных приложений без использования браузера).

Лабораторная работа 9: «Современные криптографические технологии защиты информации»[1, 2]

Задание 1. Защита авторских прав [коэффициент сложности равен 1]

Изучите и опишите применение технологии Blockchain для опеспечения владения NFT (от англ. *Non-Fungible Token*, уникальный токен).

Задание 2. Исполнение обязательств контракта [коэффициент сложности равен 2]

Изучите и опишите применение технологии Blockchain для обеспечения выполнения смарт-контракта (от англ. *Smart contract*, умный контракт).

Задание 3. Онлайн-голосование [коэффициент сложности равен 3]

Изучите и опишите применение технологии Blockchain и гомоморфных шифров для обеспечения безопасности и анонимности онлайн-голосования.

[1] Необходимо выполнить и защитить одно из предложенных заданий. Решение нужно сопроводить содержательным отчетом (в произвольной форме).

[2] При необходимости использовать криптографические средства защиты информации следует ограничиться вызовом утилиты `openssl`.