### Введение

Прохождение практики осуществлялось на кафедре СурГУ. Сургутский государственный университет основан в мае 1993 года. Основная цель развития СурГУ — трансформация в университет нового типа, поддерживающий и обеспечивающий инновационное экономическое развитие региона посредством концентрации интеллектуального потенциала, современной научно-образовательной инфраструктуры и их приближения к потребностям регионального бизнеса.

В университете реализуются программы высшего образования всех уровней по широкому спектру направлений и специальностей на базе шести учебно-научных институтов: гуманитарного образования и спорта, государства и права, естественных и технических наук, экономики и управления, медицинского и политехнического институтов. Внедряются принципы непрерывного образования, выстроены учебные кластеры «школа – колледж – вуз – предприятие». Осуществляется профессиональная переподготовка и повышение квалификации специалистов.

Приоритетными направлениями научно-исследовательской деятельности университета являются: IT-технологии, химия нефти, природопользование, экология, медицина Севера.

IT-технологии с каждым днем становятся более популярными и востребованными, одну из важнейших функций, которую должны выполнять современные информационные системы это обеспечение надежное и целостная передача информации между клиентом и сервером.

Информация — это одна из самых ценных вещей в современной жизни. Появление глобальных компьютерных сетей сделало простым получение доступа к информации как для отдельных людей, так и для больших организаций. Но легкость и скорость доступа к данным с помощью компьютерных сетей, таких как Интернет, также сделали значительными следующие угрозы безопасности данных при отсутствии мер их защиты:

* Неавторизованный доступ к информации
* Неавторизованное изменение информации
* Неавторизованный доступ к сетям и другим сервисам
* Другие сетевые атаки, такие как повтор перехваченных ранее транзакций и атаки типа "отказ в обслуживании"

Криптосистема работает по определенной методологии (процедуре). Она состоит из: одного или более алгоритмов шифрования (математических формул); ключей, используемых этими алгоритмами шифрования; системы управления ключами; незашифрованного текста; и зашифрованного текста (шифр текста).

Согласно методологии сначала к тексту применяются алгоритм шифрования и ключ для получения из него шифр текста. Затем шифр текст передается к месту назначения, где тот же самый алгоритм используется для его расшифровки, чтобы получить снова текст. Также в методологию входят процедуры создания ключей и их распространения.

В ходе практики были изучены способы симметричного и ассиметричного шифрования их отличие, преимущества и недостатки (Таблица 1, Таблица 2) каждого из них.

Таблица 1

Преимущества и недостатки симметричного шифрования данных

|  |  |
| --- | --- |
| **симметричное шифрование** | |
| **преимущества** | **недостатки** |
| скорость работы | надежность |
| требования к вычислительной мощности | относительно большое количество ключей |

Таблица 2

Преимущества и недостатки асимметричного шифрования данных

|  |  |
| --- | --- |
| **асимметричное шифрование** | |
| **преимущества** | **недостатки** |
| надежность | скорость работы |
| относительно небольшое количество ключей | требования к вычислительной мощности |
| нет необходимости передавать полный ключ | относительная сложность внесения изменений |

Была выполнена работа по написанию и внедрению блочного шифрования с использованием механизма обратной связи на языках Python, JavaScript с применением стандарта AES и метода cipher block chaining 

где:

* i - номер блока;
* k - [ключ](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F));
* IV - вектор инициализации ([синхропосылка](https://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BF%D0%BE%D1%81%D1%8B%D0%BB%D0%BA%D0%B0&action=edit&redlink=1" \o "Синхропосылка (страница отсутствует)));
* Pi - блок сообщения ([открытый текст](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D1%82%D0%B5%D0%BA%D1%81%D1%82));
* Ci-1 - зашифрованный блок ([шифротекст](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D1%82%D0%B5%D0%BA%D1%81%D1%82" \o "Шифротекст)), полученный на предыдущем шаге шифрования;
* Ek - функция, выполняющая [блочное шифрование](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D1%87%D0%BD%D0%BE%D0%B5_%D1%88%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

Расшифровка выполняется функцией {\displaystyle D\_{k}} с использованием тех же ключа *k* и вектора инициализации *IV*:



### Практическое задание

1. Изучить:

- ГОСТ 28147-89. Симметричный алгоритм блочного шифрования по стандарту AES с использованием механизма обратной связи.

2. Освоить:

- метод блочного шифрования с использованием механизма обратной связи

- разработать и протестировать программное обеспечение, которое реализует процесс шифрования и обмена данными между клиентом и сервером.

3. Подготовить и оформить отчет о прохождении производственной практики

Для выполнения поставленной задачи было решено использовать для Python модуль AES, чтобы шифровать и расшифровать данные, модуль md5 для хеширования, base64 для конвертации данных в 64 битный формат. Для заполнения по стандарту PKCS5 были написаны следующие функции:

*def pad(data):*

*# 16 - 11 = 5 // в случае больше 16 отнимаем остаток от деления на 16*

*pad = BLOCK\_SIZE - len(data) % BLOCK\_SIZE*

*# 'hello world' + 5 \* '\x05' = 'hello world\x05\x05\x05\x05\x05'*

*return data + pad \* chr(pad)*

*def unpad(padded):*

*# 'hello world\x05\x05\x05\x05\x05'*

*padded = padded.decode("utf-8")*

*# '\x05'*

*pad = padded[-1]*

*# 'hello world'*

*return padded.replace(pad, '')*

И функции \_encrypt, \_decrypt, для шифровки и расшифровки соответственно.

*def* \_encrypt(data, nonce, password):

    """Фуникция для шифровки текста для этого передайте в

    качестве параметра функции (текст, nonce, пароль)"""

*# используем стандарт md5 для хеширования*

    m = md5()

*# 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'*

*# -> b'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'*

*# хешируем пароль*

    m.update(password.encode("utf-8"))

*# берем 16-ые цисла из хеша и инициализруем key*

    key = m.hexdigest()

*# ниже можно было бы m.update(key.encode("utf-8"))*

*# для для понимания оставим так*

    m = md5()

*# 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa' + '5eca9bd3eb07c006cd43ae48dfde7fd3'*

*# b'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa5eca9bd3eb07c006cd43ae48dfde7fd3'*

*# хеш по итогу получится так же как и до этого 32 битный*

    m.update((password + key).encode("utf-8"))

*# берем 16-ые цисла из хеша и инициализруем iv*

    iv = m.hexdigest()

*# 'hello world\x05\x05\x05\x05\x05'*

    data = pad(data)

*# b'5eca9bd3eb07c006cd43ae48dfde7fd3'*

    key = key.encode("utf-8")

*# b'7d90268e5545475722b7024fdb80cfa6'*

    iv = iv.encode("utf-8")

*# b'hello world\x05\x05\x05\x05\x05'*

    data = data.encode("utf-8")

*# инициализируем класс AES*

*# (b'5eca9bd3eb07c006cd43ae48dfde7fd3', метод, b'7d90268e55454757')*

    aes = AES.new(key, AES.MODE\_CBC, iv[:16])

*# получаем батовый шифр b'R\x16\xd5HX\xd5\xb7\xe8"ua\x06\xb2y\xaaT'*

    encrypted = aes.encrypt(data)

*# конвертируем в 64-ох битный формат*

*# b'UhbVSFjVt+gidWEGsnmqVA=='*

*return* base64.urlsafe\_b64encode(encrypted)

*def* \_decrypt(edata, nonce, password):

    """Фуникция для расшифровки текста для этого передайте в

    качестве параметра функции (зафированный текст, nonce, пароль)"""

*# получаем из 64-ох битного формата байтовый шифр*

*# b'R\x16\xd5HX\xd5\xb7\xe8"ua\x06\xb2y\xaaT'*

    edata = base64.urlsafe\_b64decode(edata)

*# используем md5 стандарт для хеширования*

    m = md5()

*# 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'*

*# -> b'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'*

*# хешируем пароль*

    m.update(password.encode("utf-8"))

*# берем 16-ые цисла из хеша и инициализруем key*

*# '5eca9bd3eb07c006cd43ae48dfde7fd3'*

    key = m.hexdigest()

*# ниже можно было бы m.update(key.encode("utf-8"))*

*# для для понимания оставим так*

    m = md5()

*# 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa' + '5eca9bd3eb07c006cd43ae48dfde7fd3'*

*# b'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa5eca9bd3eb07c006cd43ae48dfde7fd3'*

*# хеш по итогу получится так же как и до этого 32 битный*

    m.update((password + key).encode("utf-8"))

*# берем 16-ые цисла из хеша и инициализруем iv*

*# '7d90268e5545475722b7024fdb80cfa6'*

    iv = m.hexdigest()

*# b'5eca9bd3eb07c006cd43ae48dfde7fd3'*

    key = key.encode("utf-8")

*# b'7d90268e5545475722b7024fdb80cfa6'*

    iv = iv.encode("utf-8")

*# инициализируем класс AES*

*# (b'5eca9bd3eb07c006cd43ae48dfde7fd3', метод, b'7d90268e55454757')*

    aes = AES.new(key, AES.MODE\_CBC, iv[:16])

*# 'hello world'*

*return* unpad(aes.decrypt(edata))

Чтобы реализовать данный алгоритм на JavaScript, так как он используется на стороне клиента, а это мобильное приложение на React Native, библиотека для crypto, которая является аналогом модуля AES для JavaScript, чтобы перенести библиотеку с Node js на React Native проделали следующие действия:

* Устанавливаем модуль browserify при помощи команды: npm install -g browserify
* Создаем файл с любым именем, к примеру crypto-in.js в который импортируем модуль crypto и сразу экспортируем при помощи команды:
* var crypto = require("crypto");
* module.exports = crypto;
* Выполняем команду browserify crypto-in.js -o crypto.js, данная команда создаст отдельный пакет JavaScript, то есть появится файл crypto.js в, которой будет 21 500 строк кода, это код из того самого модуля crypto для Node js, далее указываем внизу файла на его экспорт командой: module.exports = crypto;
* Перекидываем файлы в корневой каталог проекта React Native импортируем и все готов.

Так как модуль crypto реализует стандарт PKCS5 нам остается только написать функции аналогичные Python encrypt и decrypt, соответственно:

*const* encrypt = (input, password, callback) *=>* {

*/\*Фуникция для шифровки текста для этого передайте в*

*качестве параметра функции (текст, nonce, пароль)\*/*

*// используем стандарт md5 для хеширования*

*let* m = crypto.createHash('md5');

*// 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'*

*// -> b'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'*

*// хешируем пароль*

   m.update(password)

*// берем 16-ые цисла из хеша и инициализруем key*

*let* key = m.digest('hex');

*// ниже можно было бы m.update(key.encode("utf-8"))*

*// для для понимания оставим так*

   m = crypto.createHash('md5');

*// 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa' + '5eca9bd3eb07c006cd43ae48dfde7fd3'*

*// b'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa5eca9bd3eb07c006cd43ae48dfde7fd3'*

*// хеш по итогу получится так же как и до этого 32 битный*

   m.update(password + key)

*// берем 16-ые цисла из хеша и инициализруем iv*

*let* iv = m.digest('hex');

*// 'hello world'*

*let* data = Buffer.from(input, 'utf8').toString('binary');

*// инициализируем класс Cipheriv*

*// (метод, b'5eca9bd3eb07c006cd43ae48dfde7fd3', b'7d90268e55454757')*

*let* cipher = crypto.createCipheriv('aes-256-cbc', key, iv.slice(0, 16));

*// UPDATE: crypto changed in v0.10*

*// https://github.com/joyent/node/wiki/Api-changes-between-v0.8-and-v0.10*

*// узнаем текущую версию узла*

*let* nodev = process.version.match(/*^*v(\d+)\.(\d+)/);

*let* encrypted;

*// 'R\x16ÕHXÕ·è"ua\x06²yªT'*

*// для старых версий*

*if* (nodev[1] === '0' && parseInt(nodev[2]) < 10) {

      encrypted = cipher.update(data, 'binary') + cipher.final('binary');

*// для новых версий*

   } *else* {

      encrypted = cipher.update(data, 'utf8', 'binary') + cipher.final('binary');

   }

*// 'UhbVSFjVt+gidWEGsnmqVA=='*

*let* encoded = Buffer.from(encrypted, 'binary').toString('base64');

*// вызов функции с передачей в качестве параметра шифр*

*// 'UhbVSFjVt+gidWEGsnmqVA==', в данном случае вызов разшифровки*

   callback(encoded);

};

*const* decrypt = (input, password, callback) *=>* {

*/\*Фуникция для расшифровки текста для этого передайте в*

*качестве параметра функции (зафированный текст, nonce, пароль)\*/*

*// (безопасный 64 битный формат) в 64 битный формат, при необходимости*

   input = input.replace(/\-/g, '+').replace(/\_/g, '/');

*// получаем из 64-ох битного формата байтовый шифр*

*// b'R\x16\xd5HX\xd5\xb7\xe8"ua\x06\xb2y\xaaT'*

*let* edata = Buffer.from(input, 'base64').toString('binary')

*// используем md5 стандарт для хеширования*

*let* m = crypto.createHash('md5');

*// 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'*

*// -> b'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'*

*// хешируем пароль*

   m.update(password)

*// берем 16-ые цисла из хеша и инициализруем key*

*// '5eca9bd3eb07c006cd43ae48dfde7fd3'*

*let* key = m.digest('hex');

*// ниже можно было бы m.update(key.encode("utf-8"))*

*// для для понимания оставим так*

   m = crypto.createHash('md5');

*// 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa' + '5eca9bd3eb07c006cd43ae48dfde7fd3'*

*// b'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa5eca9bd3eb07c006cd43ae48dfde7fd3'*

*// хеш по итогу получится так же как и до этого 32 битный*

   m.update(password + key)

*// берем 16-ые цисла из хеша и инициализруем iv*

*// '7d90268e5545475722b7024fdb80cfa6'*

*let* iv = m.digest('hex');

*// инициализируем класс Decipheriv*

*// (метод, b'5eca9bd3eb07c006cd43ae48dfde7fd3', b'7d90268e55454757')*

*let* decipher = crypto.createDecipheriv('aes-256-cbc', key, iv.slice(0, 16));

*// UPDATE: crypto changed in v0.10*

*// https://github.com/joyent/node/wiki/Api-changes-between-v0.8-and-v0.10*

*let* nodev = process.version.match(/*^*v(\d+)\.(\d+)/);

*let* decrypted, plaintext;

*// для старых версий*

*if* (nodev[1] === '0' && parseInt(nodev[2]) < 10) {

      decrypted = decipher.update(edata, 'binary') + decipher.final('binary');

      plaintext = Buffer.from(decrypted, 'binary').toString('utf8');

*// для новых версий*

   } *else* {

      plaintext = (decipher.update(edata, 'binary', 'utf8') + decipher.final('utf8'));

   }

*// 'hello world'*

*// вызов функции с передачей в качестве параметра текст*

   callback(plaintext);

};

Разработка велась в текстовом редакторе Visual Studio Code, работа с сервером осуществлялась при помощи строенного плагина с ftp доступом, а также использовалась программа FileZilla для загрузки модулей на сервер, для работы с терминалом применял программу PuTTY и свое декстоп приложение для автоматизации работы, связанной с настройкой сервера, активации виртуальной среды, авторизации, открытии доступа по протоколу ssh.

В результате удалось написать пакет, который позволяет посредством шифрования используя в качестве ключа access token и refresh token осуществить обмен данными между клиентом (мобильное приложение) и сервером.

### Заключение

В ходе практики мной был изучен и применен симметричный алгоритм блочного шифрования по стандарту AES с использованием механизма обратной связи на двух языках программирования, произведено сравнение с симметричного шифрования с ассиметричным. Разработано и протестировано система, которое реализует процесс шифрования и обмена данными между клиентом и сервером. Знания и опыт полученные в ходе практик дали возможность понять, что из себя представляют криптография и как можно его применить, в дальнейшем при реализации.

### Литература

1. Айвазян С.А., Мхитарян В.С. Прикладная статистика. Основы эконометрики. Теория вероятностей и прикладная статистика. - М. : Юнита-Дана, 2001. - 656 с.
2. О. С. Зензин, М. А. Иванов. Стандарт криптографической защиты - AES (Advanced Encryption Standart) - М. : КУДИЦ - Образ, 2002. – 176 c.
3. Алгоритм шифрования AES [Электронный ресурс]. – Режим доступа: <http://book.itep.ru/6/aes.htm> (дата обращения: 13.07.2022)
4. ГОСТ 20400-2013 Продукция мебельного производства. Термины и определения [Электронный ресурс]. – Режим доступа: https://docs.cntd.ru/document/1200107173 (дата обращения: 01.05.2022)