**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский Авиационный Институт»**

**(Национальный Исследовательский Университет)**

**Институт: №8 «Информационные технологии   
и прикладная математика»   
Кафедра: 806 «Вычислительная математика   
и программирование»**

Курсовой Проект  
по предмету «Криптография»

Группа: М8О-309Б-22

Студент: О. С. Концебалов

Преподаватель: А. В. Борисов

Оценка:

Дата: 22.05.2025

Москва, 2025

**ОГЛАВЛЕНИЕ**

[1 Тема 3](#_Toc158983147)

[2 Задание 3](#_Toc158983148)

[3 Теория 4](#_Toc158983149)

[4 Ход лабораторной работы 5](#_Toc158983150)

[5 Выводы 6](#_Toc158983151)

# **Тема**

Аутентификация с асимметричными алгоритмами шифрования в сети Интернет.

# **Задание**

1. Выбрать не менее 3-ёх web-серверов сети Интернет различной организационной и государственной принадлежности.
2. Запустить Wireshark/tcpdump в режиме записи.
3. Используя Firefox/Chrome/Safari/ИнойБраузер установить https соединение с выбранным сервером и убедиться в установке соединения.
4. Сохранить данные необходимы для последующего сравнительного анализа:

* Имя сервера, его характеристики.
* Версия TLS.
* Выбранные алгоритмы шифрования.
* Полученный сертификат: версия, действителен ли сертификат, правильность ключа, удостоверяющий центр.
* Время установки соединения (от ClientHello до Finished)

1. Если список исследуемых серверов не исчерпан выбрать другой сервер и повторить соединение.
2. Если браузер поддерживал соединение TLS 1.2 / 1.3 принудительно изменить параметры TLS (для соединения в Firefox на TLS 1.0 / 1.1 в браузере перейти по адресу “about:config” и изменить раздел SSL\TLS, security.tls.version.enable-deprecated) и провести попытки соединения с выбранными серверами.
3. Провести сравнительный анализ полученной информации.
4. В качестве отчета представить результаты сравнительного анализа, выводы в отношении безопасности и корректности настройки веб-серверов с учетом их организационной игосударственной принадлежности

# **Теория**

TLS ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) transport layer security — Протокол защиты транспортного уровня), как и его предшественник [SSL](https://ru.wikipedia.org/wiki/SSL) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) secure sockets layer — слой защищённых сокетов), — [криптографические протоколы](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB), обеспечивающие защищённую передачу данных между узлами в сети [Интернет](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82). TLS и SSL используют [асимметричное шифрование](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D0%B8%D0%BC%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D0%B0%D1%8F_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F) для аутентификации, [симметричное шифрование](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D1%8B%D0%B5_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B) для конфиденциальности и [коды аутентичности сообщений](https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%B8%D1%82%D0%BE%D0%B2%D1%81%D1%82%D0%B0%D0%B2%D0%BA%D0%B0) для сохранения целостности сообщений.

#### **Подтверждение связи с аутентификацией клиента**

В данном примере показана полная аутентификация клиента (в дополнение к аутентификации сервера, как в предыдущем примере) с помощью обмена сертификатами между сервером и клиентом.

1. Фаза переговоров:
   * Клиент посылает сообщение ClientHello, указывая последнюю версию поддерживаемого TLS-протокола, случайное число и список поддерживаемых методов шифрования и сжатия, подходящих для работы с TLS;
   * Сервер отвечает сообщением ServerHello, содержащим: выбранную сервером версию протокола, случайное число, посланное клиентом, подходящий алгоритм шифрования и сжатия из списка, предоставленного клиентом;
   * Сервер посылает сообщение Certificate, которое содержит цифровой сертификат сервера (в зависимости от алгоритма шифрования этот этап может быть пропущен);
   * Сервер посылает сообщение CertificateRequest, которое содержит запрос сертификата клиента для взаимной проверки подлинности;
   * Клиент посылает сообщение Certificate, которое содержит цифровой сертификат клиента;
   * Сервер отсылает сообщение ServerHelloDone, идентифицирующее окончание подтверждения связи;
   * Клиент отвечает сообщением ClientKeyExchange, которое содержит открытый ключ PreMasterSecret или ничего (опять же зависит от алгоритма шифрования);
   * Клиент и сервер, используя ключ PreMasterSecret и случайно сгенерированные числа, вычисляют общий секретный ключ. Вся остальная информация о ключе будет получена из общего секретного ключа (и сгенерированных клиентом и сервером случайных значений);
2. Клиент посылает сообщение ChangeCipherSpec, которое указывает на то, что вся последующая информация будет зашифрована установленным в процессе подтверждения связи алгоритмом, используя общий секретный ключ. Это сообщения уровня записей и поэтому имеет тип 20, а не 22;
   * Клиент посылает сообщение Finished, которое содержит хеш и MAC, сгенерированные на основе предыдущих сообщений процедуры подтверждения связи;
   * Сервер пытается расшифровать Finished-сообщение клиента и проверить хеш и МАС. Если процесс расшифровки или проверки не удаётся, подтверждение связи считается неудавшимся, и соединение должно быть оборвано.
3. Сервер посылает ChangeCipherSpec и зашифрованное сообщение Finished, и в свою очередь клиент тоже выполняет расшифровку и проверку.

С этого момента подтверждение связи считается завершённым, протокол установленным. Всё последующее содержимое пакетов идёт с типом 23, а все данные будут зашифрованы.

# **Ход лабораторной работы**

Для выполнения Курсового Проекта я решил выбрать следующие веб-сервера (у меня это сервера сайтов): gosuslugi.ru, github.com и steampowered.com

Первым делом мне пришлось установить Wireshark (удалил его незадолго до начала работы над КП с мыслью, что он мне больше не пригодится).

Чтобы определить имя сервера, откроем нужный сайт в браузере, затем в Wireshark в поле для фильтра введём «dns», найдём пакет с именем хоста нужного нам сервера, откроем параметры этого пакета, имя находится в Domain Name System (response) → Quieries

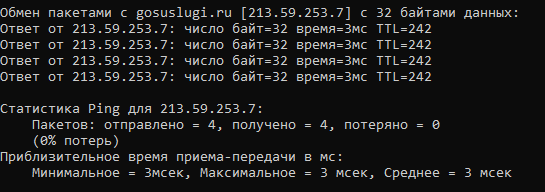
Чтобы определить версию TLS и алгоритмы шифрования, в фильтре для пакетов указываем «tls», откроем параметры этого пакета, откроем Transport Layer Security, там можно увидеть версию TLS, затем в разделе Cipher Suites будут перечислены используемые алгоритмы шифрования

Чтобы получить информацию о сертификате, находим пакет с сообщением ServerHello с помощью фильтра «ssl.handshake.type == 2», откроем параметры этого сертификата, перейдём по Transport Layer Security и увидим несколько веток, отвечающих за сертификат.

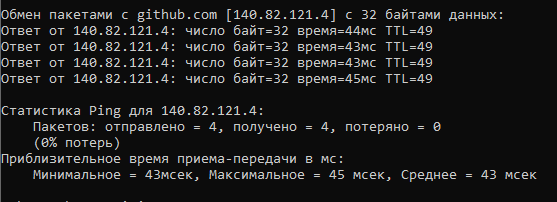
Чтобы определить время установки соединения от ClientHello до Finished, находим пакет с ClientHello, следуем по потоку TCP до пакета с Finished и находим разницу во времени между этими двумя пакетами.

В начале столкнулся с проблемой – как мне отловить трафик только от нужных сайтов? После недолгих раздумий пришел к решению – пингануть нужные сайты и узнать их IP-адреса, а потом задать их в CaptureFilters

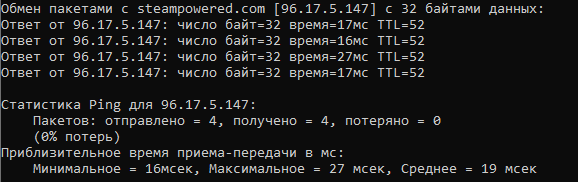
gosuslugi.ru (213.59.254.7)



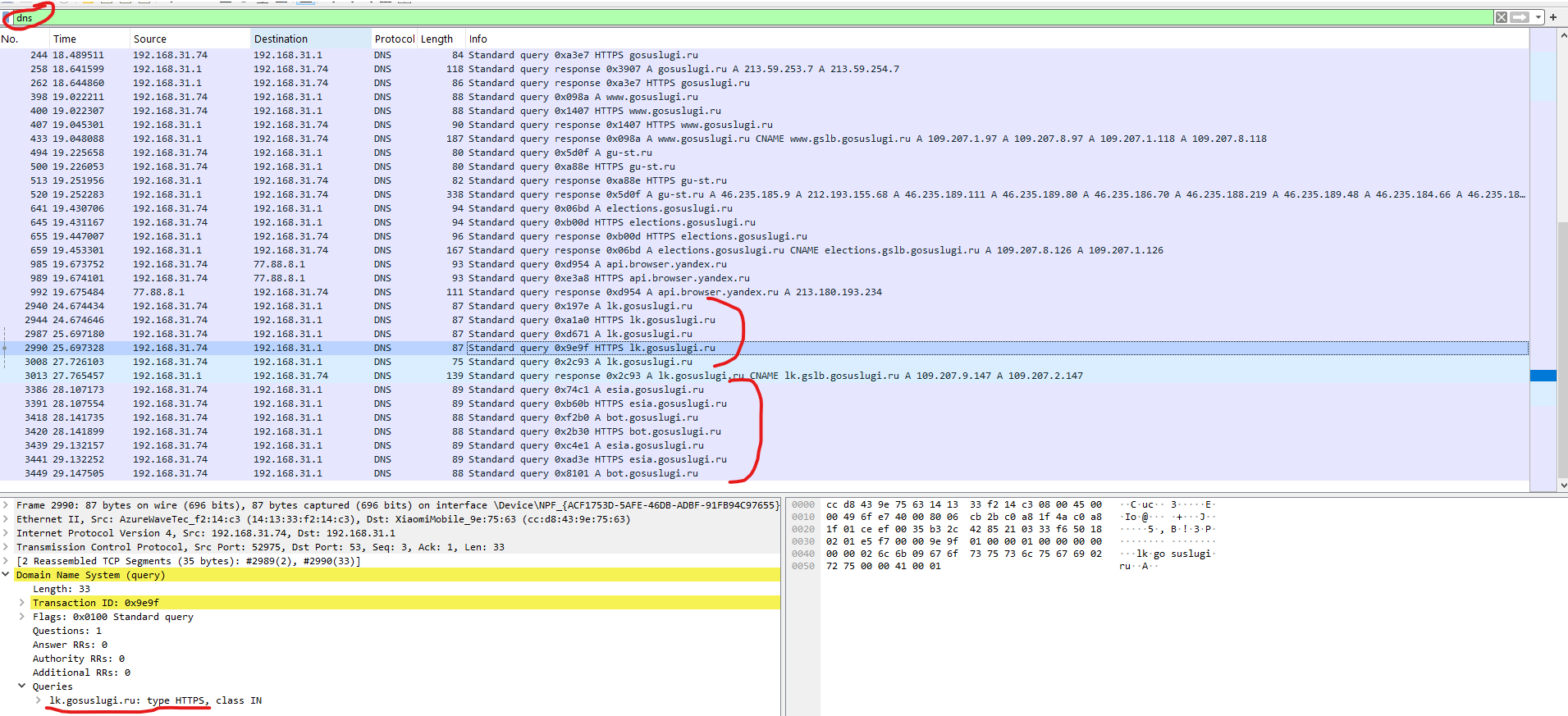
github.com (140.82.121.4)



steampowered.com (96.17.5.147)



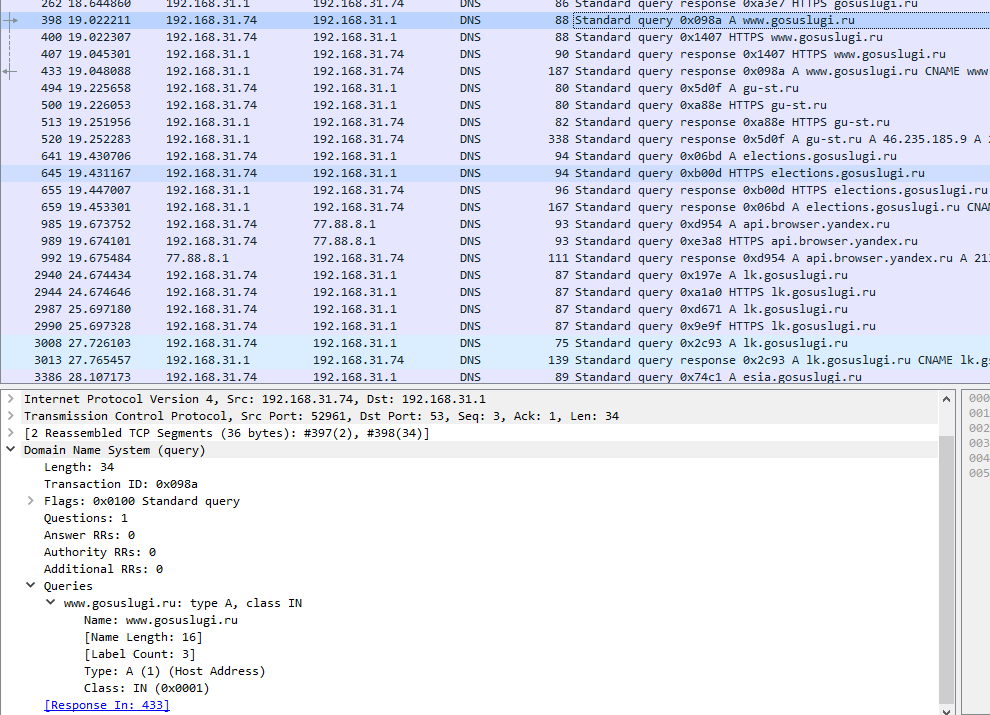
Но это не помогло – IP выдавались динамически и изменялись. Тогда решил захватывать весь трафик на протяжении небольшого промежутка времени, параллельно активно работая (посылая запросы на нужный сайт), потом останавливать захват трафика и фильтровать его, на основании пунктов, приведенных выше. Такой способ сработал.



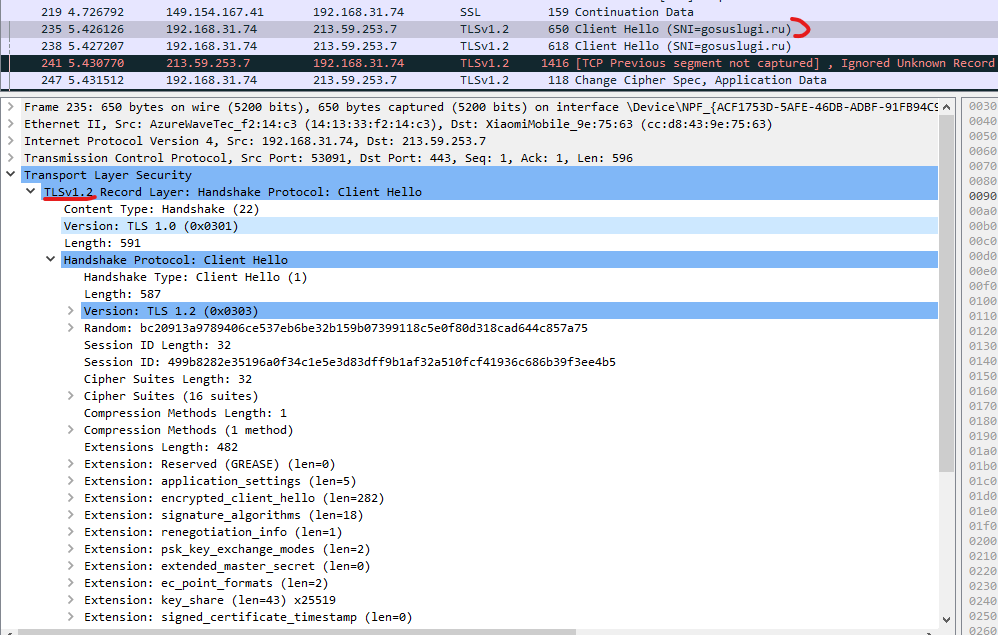
В итоге для выбранных сайтов получил следующие результаты:

***gosuslugi.ru***

1) Имя сервера: gosuslugi.ru

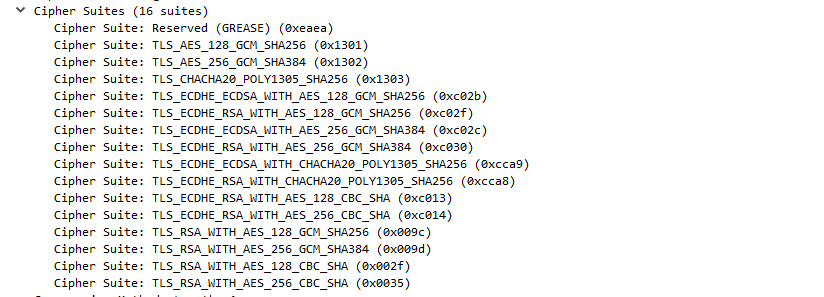


2) Версия TLS: 1.2



3) Алгоритмы шифрования: TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256 (0xc02f)

Были предложены 16 вариантов



4) Сертификат: 00:9B:C6:EB:61:E1:1C:C3:08:10:FD:CB:86:7D:9D:21:F5; версия 3 (2),

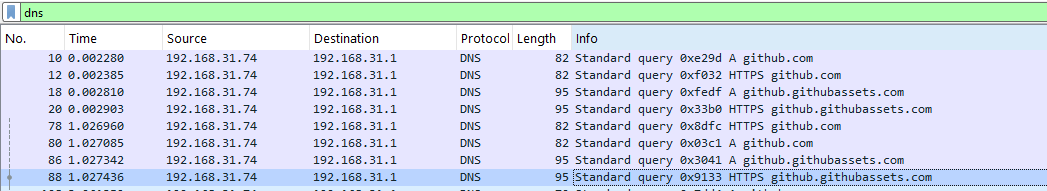
Удостоверяющий центр: rdnSequence: 3 items (id-at-commonName=GlobalSign GCC R3 DV TLS CA 2020,id-at-organizationName=GlobalSign nv-sa,id-at-countryName=BE)

5) Время установки соединения: 8,11 мс, что свидетельствует о быстром процессе установления защищённого соединения

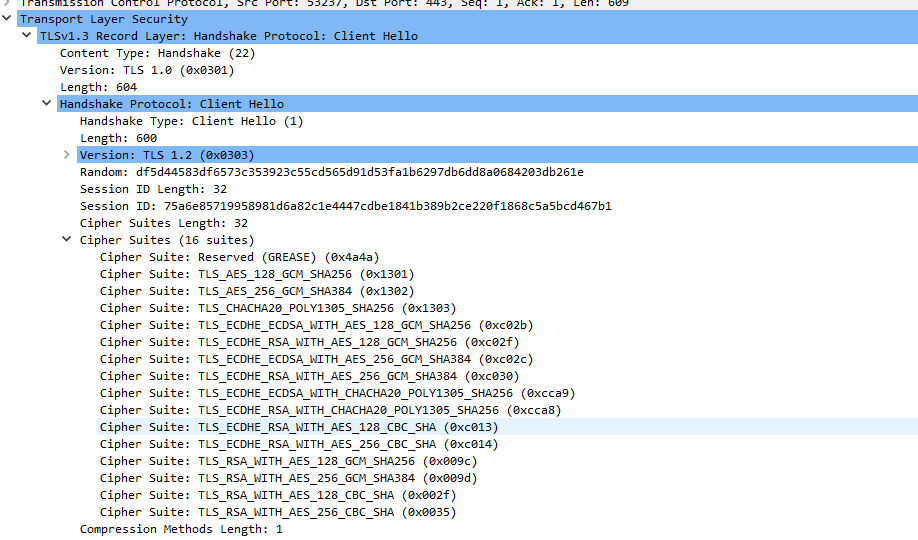
При изменении на версию TLS 1.0 сайт госуслуг упал и появилось окно с предупреждением.

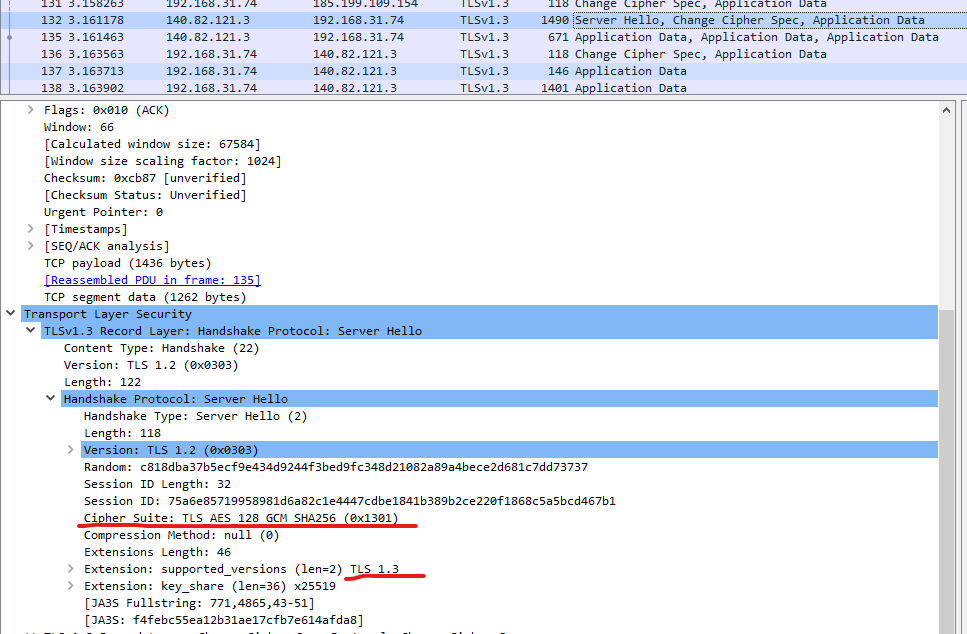
***github.com***

1) Имя сервера: github.com



2) Версия TLS: 1.2





3) Алгоритмы шифрования: предлагаются все те же 16 вариантов для шифрования, был выбран

Cipher Suite: TLS\_AES\_128\_GCM\_SHA256 (0x1301)

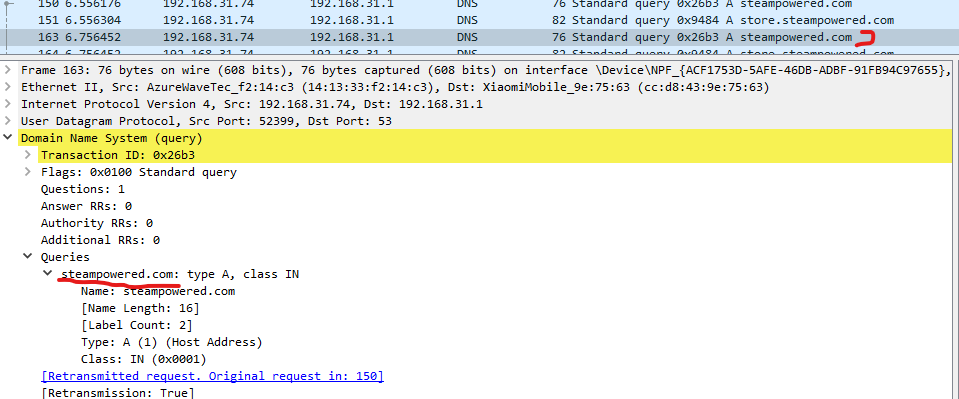
4) Сертификат: версия 3, 0C:D0:A8:BE:C6:32:CF:E6:45:EC:A0:A9:B0:84:FB:1C



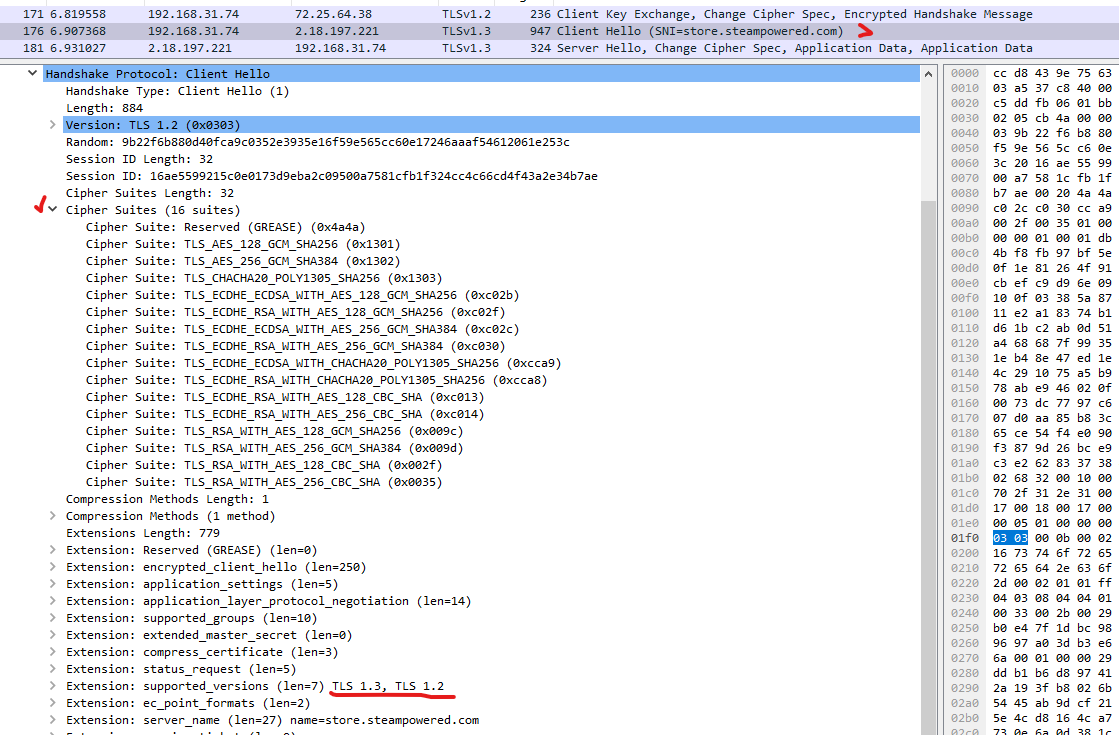
5) Время установки соединения: 1,93 с, могу предположить, что такое длительное время связано с проблемами интернета в общежитии – произошел пролаг или было нестабильное соединение и время ожидания выросло

***steampowered.com***

1) Имя сервера: steampowered.com



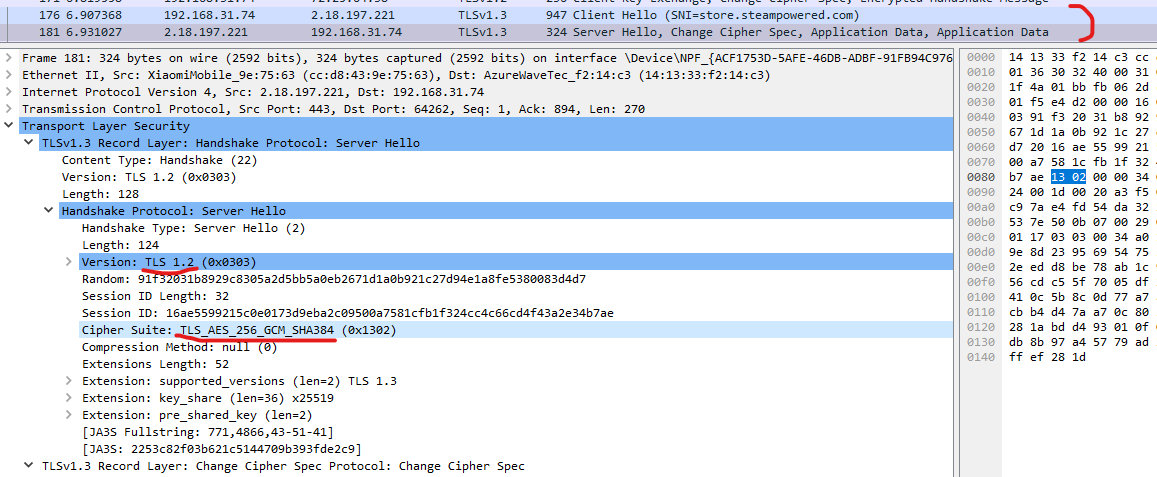
2) Версия TLS: 1.2, поддерживается 1.3



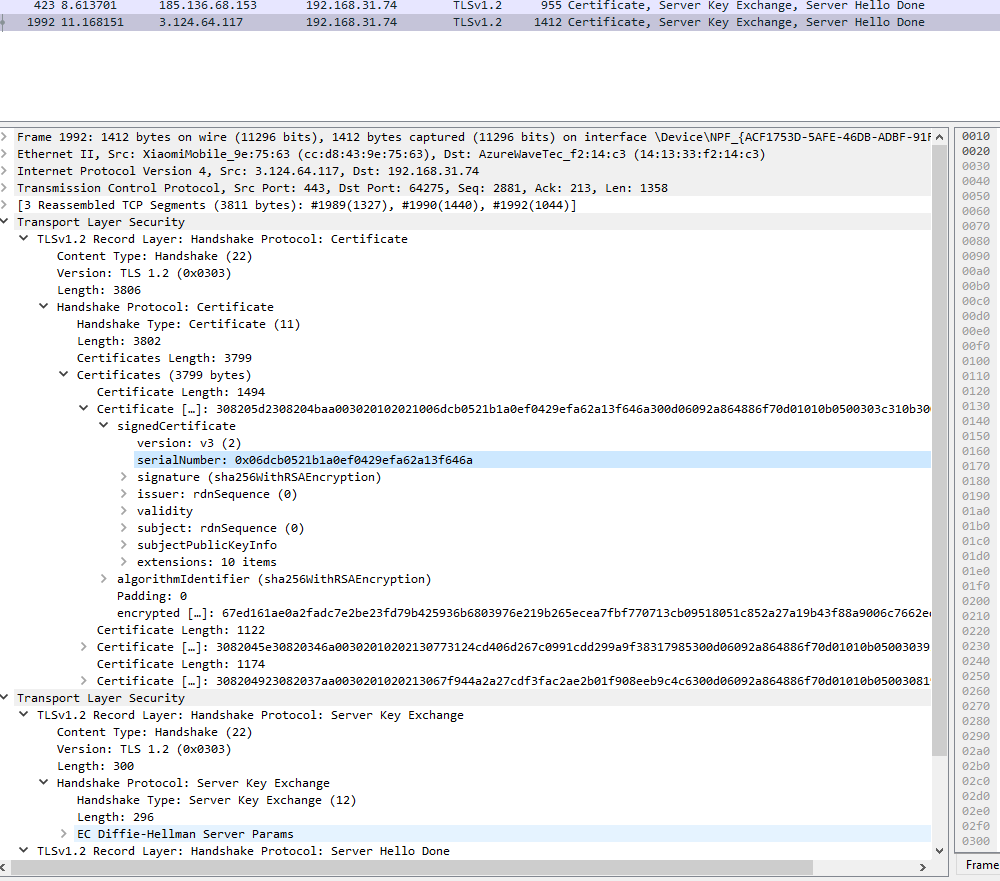
3) Алгоритмы шифрования: аналогично с другими сайтами были предложены 16 вариантов, из них был выбран

Cipher Suite: TLS\_AES\_256\_GCM\_SHA384 (0x1302)

Так же видно, что для установки соединения была выбрана версия TLS 1.2



4) Сертификат: версия 3, 03:FC:D4:17:85:00:A1:98:3F:EA:68:BE:D8:2B:AC:F9:FD:DE



5) Время установки соединения: 0.718 с

Из собранных характеристик видно, что все сервера, работают с TLSv1.2 и поддерживают TLSv1.3. Принудительно отключим версии TLS новее 1.0 и посмотрим, что произойдём, для этого перейдём в Firefox по адресу about:config и установим в поле security.tls.version.max значение 1. Попробуем снова открыть сайты. Попытки установить соединение с любым из серверов, приводят к возникновению ошибки SSL\_ERROR\_PROTOCOL\_VERSION\_ALERT.

# **Выводы**

В ходе выполнения курсового проекта я приобрел навыки работы и понимания асимметричных алгоритмов шифрования. Они используются для защиты, безопасности данных в сети и аутентификации пользователей в сети, в различных серверах, системах. Так же я изучил часто используемые мной веб-сайты с точки зрения их защищенности и понял, все крупные современные сайты используют новейшие алгоритмы шифрования и защищенный протокол https.

Так же в ходе КП научился базово пользоваться Wireshark’ом, до этого использовал его только чтобы бегло посмотреть на пакеты, когда изучал Компьютерные Сети. Оказалось, что это довольно мощный инструмент, который может пригодится в разных задачах, например, для траблшутинга сетевых проблем на работе

# **Список используемой литературы**

* <https://ru.wikipedia.org/wiki/TLS>
* <https://wiki.wireshark.org/TLS>
* https://www.wireshark.org/docs/