

Сеть и сетевые протоколы: Высокоуровневые сетевые протоколы



Александр
Гришин



Александр Гришин

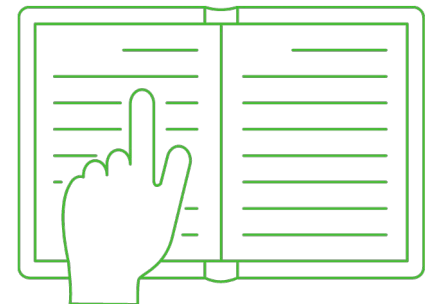
Инженер, компания YADRO

Предисловие

На этом занятии мы поговорим о:

- различных протоколах прикладного уровня;
- основах криптографии;
- работе сертификатов.

По итогу занятия вы получите представление о работе протоколов прикладного уровня и о шифровании данных с помощью [SSL](#) и [TLS](#).



План занятия

1. [Предисловие](#)
2. [Прикладной уровень модели OSI](#)
3. [Основы шифрования](#)
4. [SSL сертификаты](#)
5. [Letsencrypt](#)
6. [Итоги](#)
7. [Домашнее задание](#)



Прикладной уровень модели OSI

Прикладной уровень OSI

Прикладной уровень – последний уровень модели OSI, обеспечивает взаимодействие сети и пользователя.

Примеры протоколов:

- HTTP;
- DNS;
- IMAP;
- SSH;
- FTP.



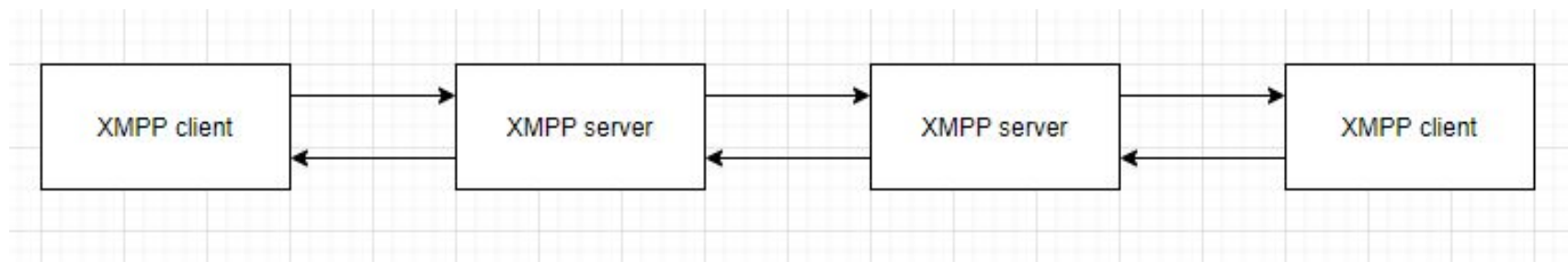
XMPP

XMPP (eXtensible Messaging and Presence Protocol) – расширяемый протокол обмена сообщениями и информацией о присутствии.

В данный момент используется в основном для **создания частных серверов**.

➔ Например в сетях без интернета.

Полный список ПО реализующий протокол доступен на [сайте](#) сообщества.



FTP

FTP (File transfer protocol) – протокол передачи файлов. Реализует обмен файлов между клиентом и сервером.

Состоит из 2-х процессов:

- управляющий процесс;
- процесс передачи данных.

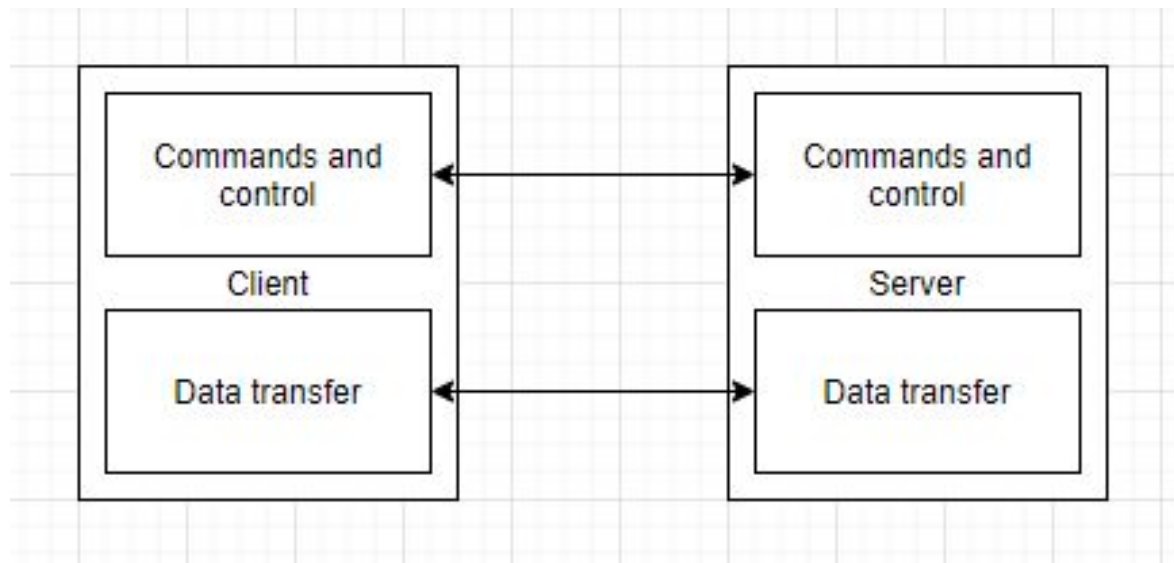
Есть поддержка активного и пассивного режима.

Список ПО реализующее данный протокол:

- Filezilla;
- WinSCP.



FTP



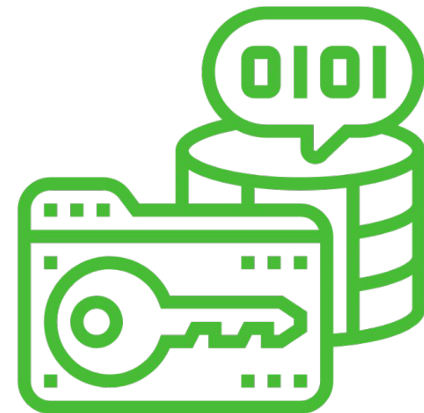
RLogin

RLogin – протокол удаленного входа в систему.

Позволяет подключаться удаленно и выполнять команды на сервере.

Основной недостаток: все данные передаются без шифрования.

➔ По этой причине был дополнен и превратился в **SSH**.





Основы шифрования

Шифрование с открытым ключом

У каждого участника есть **2 ключа**: публичный и приватный.

Публичный ключ может только шифровать сообщения.

Приватный ключ может только расшифровывать сообщение, и только те сообщения что были зашифрованы связанным с ним приватным ключом.

Асинхронное шифрование является медленным и ресурсозатратным, поэтому используется только для гарантии безопасного обмена общим сеансовым ключом.

Сеансовый ключ может зашифровывать и расшифровывать. Создается на время в момент рукопожатия.

Как работает RSA

- берем 2 простых числа:

$$p = 7; q = 11.$$

- рассчитываем произведение:

$$n = p * q = 77.$$

- вычисляем функцию Эйлера:

$$F = (p-1) * (q-1) = 6 * 10 = 60.$$

- выбираем число **e**: простое, меньше **F** и взаимно простое с ним

Например, 13.

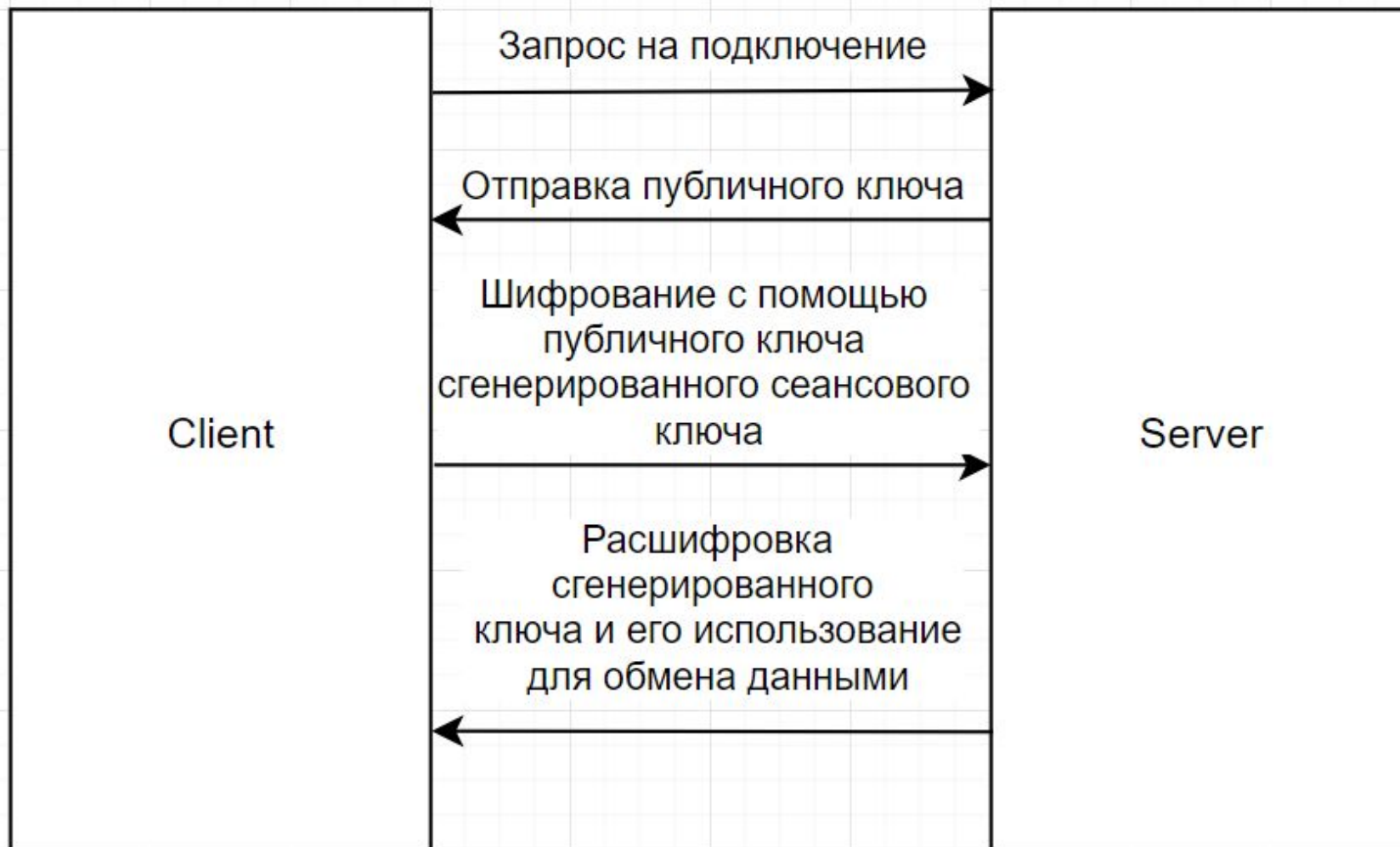
- теперь пара чисел $\{e, n\} \{13, 77\}$ — это мой открытый ключ.

- ...

Как работает RSA

- ...
- теперь нужно вычислить число d , обратное e по модулю F :
 $(d * e) \% F = 1$.
Или $(d * 13) \% 60 = 1$.
- возьмем его равным 37 .
- теперь $\{d, n\} \{37, 77\}$ – мой закрытый ключ.
- зашифруем сообщение « 50 » с помощью ключа $\{13, 77\}$.
Получаем $(50 ^ 13) \bmod 77 = 29$
- расшифруем закрытым ключом:
 $(29 ^ d) \bmod n = (29 ^ 37) \bmod 77 = 50$

SSL рукопожатие



Генерирование ключей SSH

- устанавливаем **OpenSSH** сервер (Ubuntu):

```
sudo apt-get install openssh-server
```

- генерируем ключ:

```
ssh-keygen -t rsa
```

- получаем файлы:

- `/home/user/.ssh/id_rsa;`

- `/home/user/.ssh/id_rsa.pub.`

- ...

Генерирование ключей SSH

- ...

- записываем публичный ключ:

```
cat /home/user/.ssh/id_rsa.pub >> /home/user/.ssh/authorized_keys
```

- пробуем подключиться с использованием ключа:

```
ssh -i /home/user/.ssh/id_rsa localhost
```



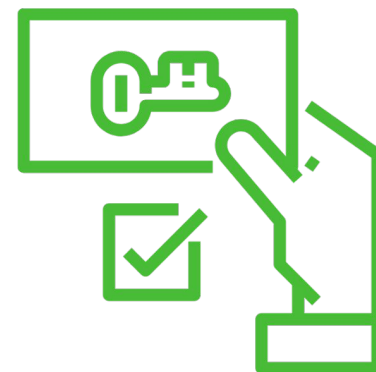
SSL Сертификаты

Цепочка доверия

Корневой сертификат – сертификат, с правом подписи. Генерируется изначально.

Промежуточный сертификат – обычно сертификат с правом подписи, подписанный корневым, выдается компаниям посредникам между ЦС и конечным пользователям.

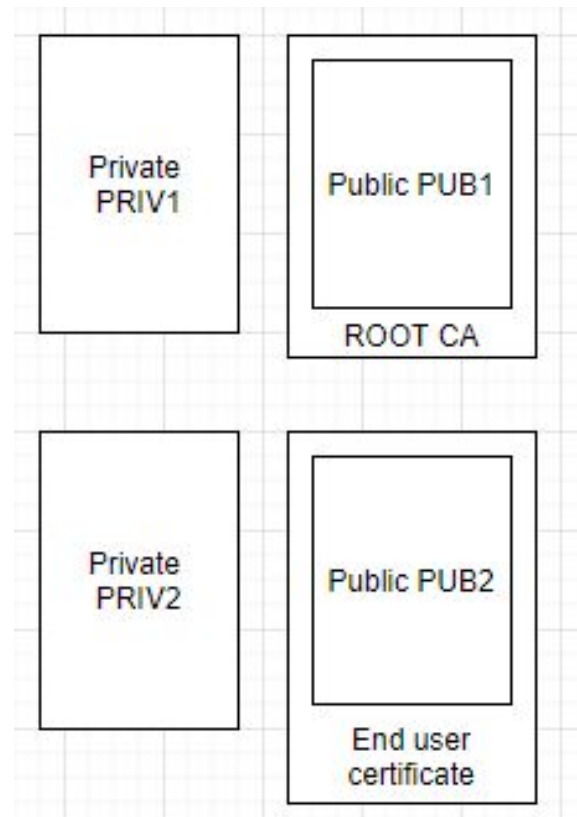
Конечный сертификат – подтверждает владение неким цифровым активом (например, доменом).



Цепочка доверия

Реализация получения конечного сертификата:

1. Публичный ключ (PUB2) отправляется в ЦС.
2. ЦС создает сертификат в котором есть зашифрованный приватным ключом PRIV1 ключ PUB2.
3. Любой, кто хочет проверить подлинность сертификата, может взять ключ PUB1, расшифровать сертификат и получить PUB2.



Работа с SSL

- `openssl` — основная утилита для работы с SSL;
- `openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365` — сгенерировать новый сертификат;
- `openssl s_client -connect ya.ru:443` — подключение к серверу и запрос на получение сертификатов.



Установка сертификата в систему

Ubuntu

- `sudo mkdir /usr/share/ca-certificates/extra`
- `sudo cp foo.crt /usr/share/ca-certificates/extra/foo.crt`
- `sudo update-ca-certificates`

CentOS

- `sudo cp foo.crt /etc/pki/ca-trust/source/anchors/foo.crt`
- `sudo update-ca-trust`

Создание своего центра сертификации

- генерируем ключ и корневой сертификат, поля в сертификате указываем любые;

```
openssl genrsa -out ca.key 2048 && openssl req -x509 -new -nodes  
-key ca.key -sha256 -days 720 -out ca.pem
```

- сразу же сделаем сертификат в форме **crt**;

```
openssl x509 -in ca.pem -inform PEM -out ca.crt
```

- установим сертификат в систему;

```
sudo cp ca.crt /usr/local/share/ca-certificates/myca.crt && sudo  
update-ca-certificates
```

- ...

Создание своего центра сертификации

- приступим к выпуску самого сертификата, генерируем ключи;
`openssl genrsa -out certificate.key 2048`
- на основе ключа создаем **CSR**;
`openssl req -new -key certificate.key -out certificate.csr`
- подписываем **CSR** нашим корневым сертификатом, тем самым создаем конечный сертификат.

```
openssl x509 -req -in certificate.csr -CA ca.pem -CAkey ca.key  
-CAcreateserial -out certificate.crt -days 360 -sha256
```

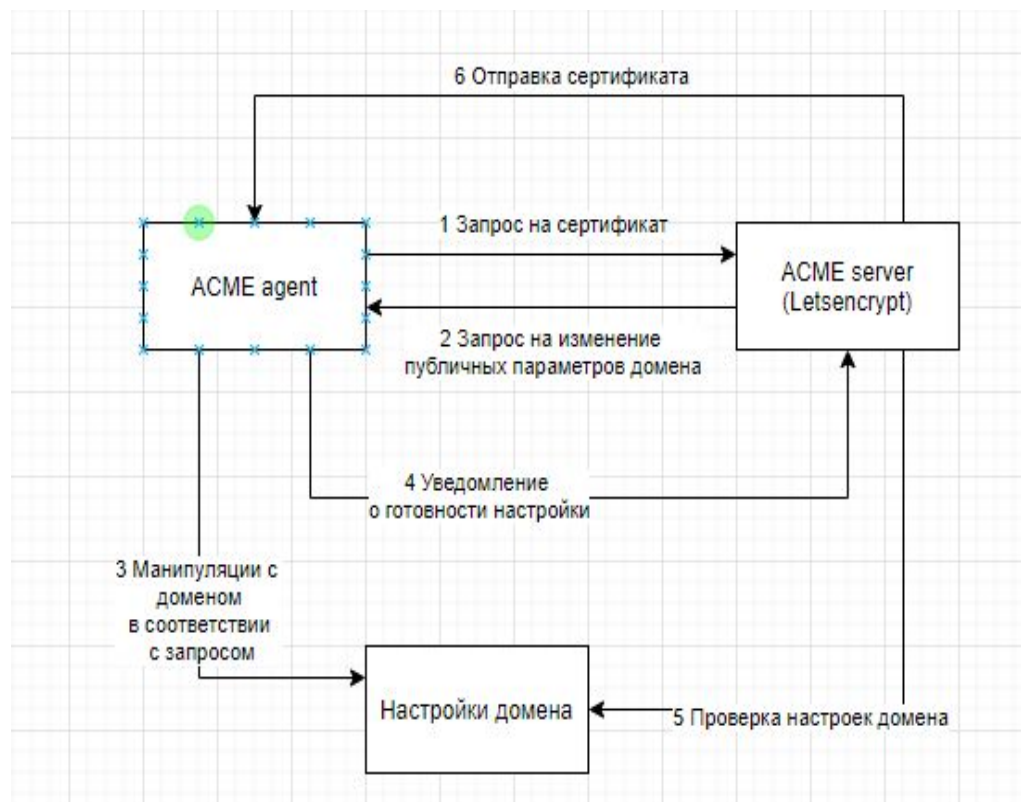



Letsencrypt

ACME

Описывает методы тестирования домена и частных / публичных ключей для запрашиваемого сертификата.

→ Позволяет автоматизировать процесс получения сертификата.



Настройка LE на работу с Nginx

- устанавливаем **Certbot**;
`sudo yum install certbot`
- устанавливаем плагин для работы с **Nginx**;
`sudo yum install python-certbot-nginx`
- проверяем конфигурацию **Nginx** и запускаем **Certbot**.
`sudo certbot --nginx -d example.com`



Итоги

Итоги

Сегодня мы рассмотрели высокоуровневые протоколы и работу SSL:

- несколько примеров протоколов;
- понимаем как работает шифрование;
- умеем подписывать сертификаты;
- понимаем способы выпуска сертификатов.



Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера .
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и
пишите отзыв о лекции!**

Александр Гришин