GeekBrains

Git. Базовый курс

# Урок 12

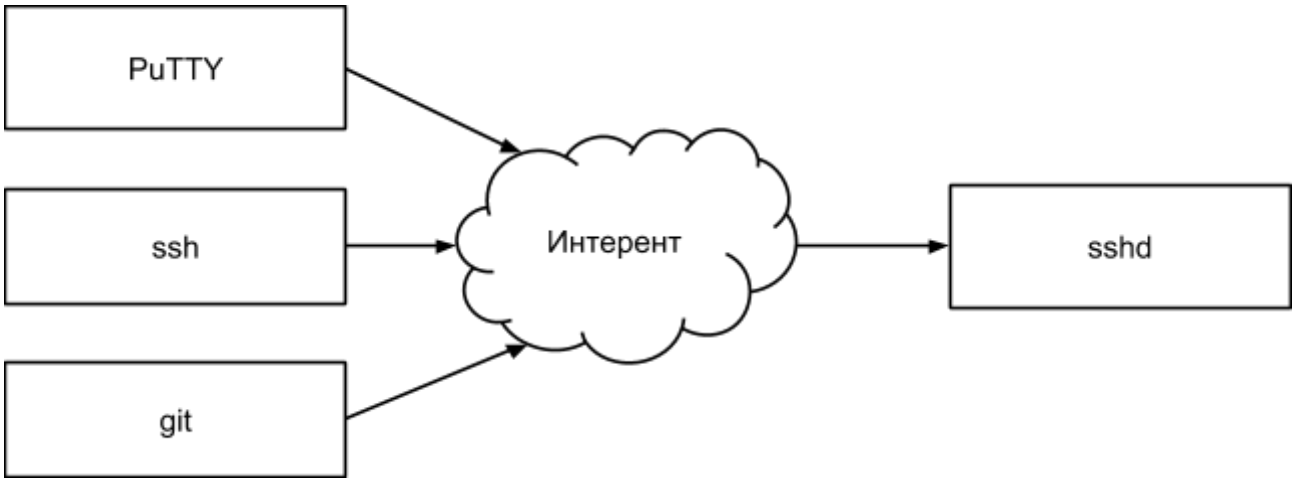
## Протокол SSH

# Протокол SSH

Когда мы взаимодействуем с удаленным git-репозиторием, у нас имеется минимум два компьютера: клиент, с которого мы отправляем команды, и удаленный сервер, который их принимает и выполняет.

Для связи с сервером используется протокол SSH, который является сокращением от Secure Shell. Основное внимание в протоколе уделяется безопасности и шифрованию передаваемых сведений. Даже если перехватить поток данных, расшифровать их почти невозможно.

Протокол является универсальным, используется не только для работы с удаленными git-репозиториями, но и для управления любыми удаленными серверами.




SSH является клиент-серверным протоколом. Предполагается, что на хосте, к которому необходимо получить доступ, запущен SSH-сервер, а на клиенте есть клиентская программа, которая к нему обращается по сети. Это может быть и SSH-клиент, и любая другая программа, которая поддерживает этот протокол, например, git-клиент.

# Соединение по протоколу SSH

Установка клиентов для различных операционных систем подробно освещается в методических материалах: (https://docs.google.com/document/d/17Lx_mavJyJxCAT230noYLkjdTmUOuKmUadCDx7jHdRU/edit#).

Будем считать, что все необходимые утилиты уже установлены. Сначала рассмотрим работу с SSH в UNIX-подобной операционной системе.

Для того, чтобы обратиться к удаленному серверу, необходимо выполнить команду ssh, передав ей адрес или домен удаленного хоста:

```
$ ssh 46.36.218.162
```

IP-адрес выше приведен для примера, чтобы получить доступ по SSH, потребуется сервер с установленным на него SSH-сервером.

При этом утилита ssh будет использовать в качестве логина имя текущего пользователя.

```
$ whoami
```

Так как имя текущего пользователя редко совпадает с именем пользователя на удаленном сервере, его можно указать явно, отделив от адреса символом собачки @:

```
$ ssh root@46.36.218.162
```

Если соединение с сервером осуществляется в первый раз, удаленный сервер может попросить подтверждение от пользователя. Получив подтверждение, он сохраняет адрес удаленного хоста в файле ~/.ssh/known_hosts.

После установки соединения и передачи на сервер логина клиенту предлагается ввести пароль (при вводе пароля символы не отображаются в консоли). Используется пароль пользователя на удаленном компьютере. Если пароль верный, выводится приглашение командной строки удаленного сервера.

Сейчас мы находимся на удаленном сервере. Давайте выполним какую-нибудь команду, например, запросим список файлов и каталогов в текущей папке:

```
$ ls -la
```

Покинуть сервер можно, введя команду exit:

```
$ exit
```

Давайте посмотрим содержимое локальной папки .ssh:

```
$ ls -la ~/.ssh/
```

Как видно, у нас появился файл known_hosts:

```
$ cat known_hosts
```

В следующий раз, обнаружив IP-адрес в этом файле, ssh-клиент сразу установит соединение с сервером без дополнительных вопросов.

Можно не устанавливать диалоговый сеанс, а передать команду для выполнения сразу после адреса. Команда будет выполнена на удаленном сервере, ее результат будет выведен в консоль, а SSH-соединение будет закрыто.

```
$ ssh root@46.36.218.162 uptime
```

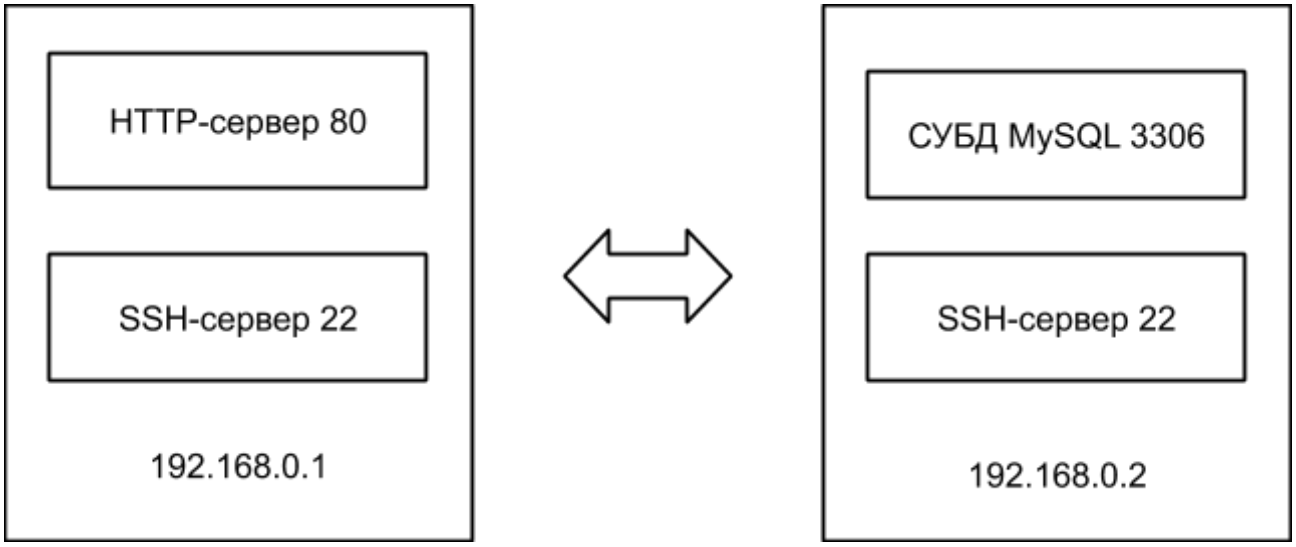
Итак, мы запросили время непрерывной работы сервера и его текущую нагрузку.

Если команда содержит пробелы, ее следует заключить в кавычки:

```
$ ssh root@46.36.218.162 "ls -la"
```

# Порт

Каждый хост в сети имеет IP-адрес, по которому один компьютер отличается от другого. Однако на одном хосте может работать несколько серверов. Чтобы их различать, используется специальный номер — порт.




Некоторые порты стандартизированы, например, 80-й порт обычно назначают HTTP-серверу, 3306 — серверу базы данных MySQL. Для SSH-сервера традиционно используется порт 22.

В целях безопасности номер порта может изменяться в настройках SSH-сервера. В этом случае в ssh-клиенте его можно указать при помощи параметра **–p**:

```
$ ssh -p 2222 user@46.36.218.162
```

Настройки SSH-клиента изменяются индивидуально для каждого пользователя, для этого прибегают к отдельному конфигурационному файлу в домашнем каталоге пользователя по пути ~/.ssh/config. В свежеустановленной системе этот файл может отсутствовать, и требуется его самостоятельное создание.

Конфигурационный файл ~/.ssh/config можно использовать для создания псевдонимов:

```
Host node1
Hostname 46.36.218.162
Port 22
User root
```

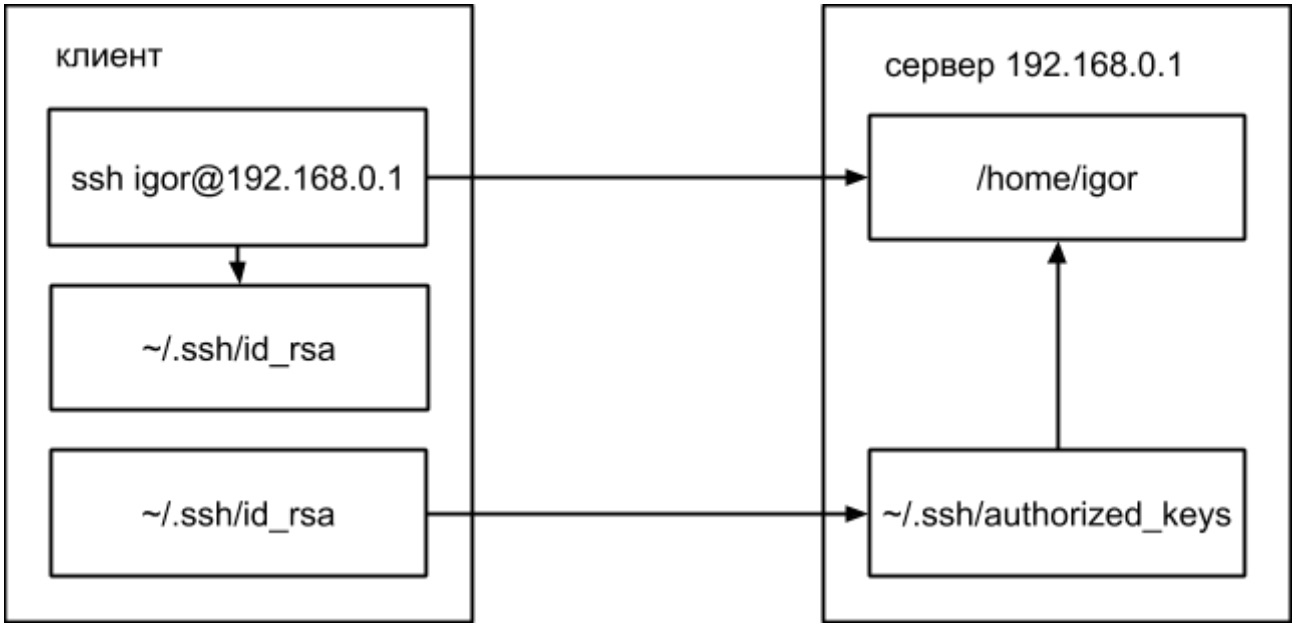
Здесь для псевдонима node1 задается адрес хоста, номер порта и имя пользователя. Благодаря этому можно использовать сокращенный вариант команды доступа на сервер:

```
$ ssh node1
```

С использованием псевдонима команда получается короче и легко запоминается.

# Доступ по ключу

Конфигурационный файл config позволяет задать большинство параметров соединения, за исключением пароля. Тем не менее, задачи, особенно те, которые выполняются автоматически в скриптах, не могут прерываться или и ожидать ввода данных от пользователя. В этом случае прибегают к доступу по открытому ключу.




Клиент заводит пару ключей: открытый и закрытый. Закрытый ключ помещается в домашнем каталоге локального компьютера ~/.ssh/id_rsa. Открытый (public) ключ id_rsa.pub размещается на сервере в конфигурационном файле ~/.ssh/authorized_keys в домашнем каталоге пользователя, под учетной записью которого осуществляется вход на сервер.

Для генерации ключей служит команда ssh-keygen, которую следует выполнить на клиентской машине:

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/igor/.ssh/id_rsa):
Created directory '/home/igor/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/igor/.ssh/id_rsa.
Your public key has been saved in /home/igor/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:1eVTqU42ZVtns6DPMN2S1y3EDdK4K+ClI8XYApzevEU igor@cruiser
The key's randomart image is:
+---[RSA 2048]----+
|   . .       .+ooo|
|      +   E  ..=+*=|
|   . + =  . +oX.O|
|      . = *..+.O.*o|
|      *S+  O.+.  |
|      o + . .+   |
|       . . .     |
|                 |
|                 |
+----[SHA256]-----+
```

Во время выполнения команда задаст несколько вопросов, в частности, она попросит указать путь, куда будут сохранены ключи (по умолчанию папка .ssh домашнего каталога пользователя), далее будет предложено ввести пароль для закрытого ключа. Настоятельно рекомендуется его указать, т. к. в случае, если ключ будет похищен злоумышленниками, им не смогут воспользоваться без пароля.

```
$ ls -la ~/.ssh
```

В результате выполнения команда создаст в домашнем каталоге скрытый подкаталог .ssh, в котором размещается закрытый (id_rsa) и открытый (id_rsa.pub) ключи.

Закрытый ключ никогда не должен попадать в чужие руки, передаваться через незащищенные сетевые каналы. В идеале он вообще не должен покидать компьютер, на котором был создан. Открытый ключ может свободно распространяться, вы можете его регистрировать на всех хостах, к которым хотите получить доступ, включая площадки хост-провайдеров и удаленные репозитории, вроде GitHub.

После того, как ключи будут сгенерированы, открытый ключ id_rsa.pub переправляется на сервер и дописывается в конец файла authorized_keys:

```
$ cd .ssh
$ cat authorized_keys
$ echo "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCk1C/2ZgrQRKlOezXSuQX9NFr1y2ctMSWHkJwrMPKXtqLO2IZuCN+py
PA2X5GDywDml/yS50QRpJhuFJdQRHeSpOEbi1vmYEDh9PWty4gtGddrBfXfwysmjCmlpgfGaUCT1zc6U1yA/V
8D79ufkfmmtGU6ozVwDJJ7obU6ruXNAqm08KRds/jXgnwr94x3QX79z+l94GiQ/M/z1O6+zeZqmLMhw76vqqn
iNjH3H+Z6cqnMurm78HwoLskbPFfl0D0dna1HkKWuHO69GT2qYt7In67ohET8/s+aBTSfEFckLz338qJnb7Xd
sn+pSc4lbwC751fLGK1EESyryo1KOUIN igorsimdyanov@MacBook-Pro-Igor.local" >>
authorized_keys
```

В случае GitHub следует перейти в настройки (Settings) в раздел SSH and GPG keys и добавить новый SSH-ключ.

Порядок работы с SSH в Windows немного отличается от работы в UNIX-системе. Изначально инструментарий для работы с протоколом SSH был разработан в UNIX-подобных операционных системах, поэтому в них все довольно единообразно. Приемы работы, которые мы рассматривали до этого момента, будут справедливы и для Linux и для MacOS. В Winodws порядок генерации ключей и доступ на сервер может отличаться.

Для SSH-доступа можно воспользоваться набором утилит PuTTY, UNIX-окружением Cygwin, в том числе тот, который поставляется совместно с Git for Windows. Начиная с Windows 10, можно воспользоваться UNIX-подсистемой, которая более подробно описывается в методических материалах. Конечно же, для запуска UNIX-окружения можно воспользоваться виртуальными машинами, например VirtualBox.

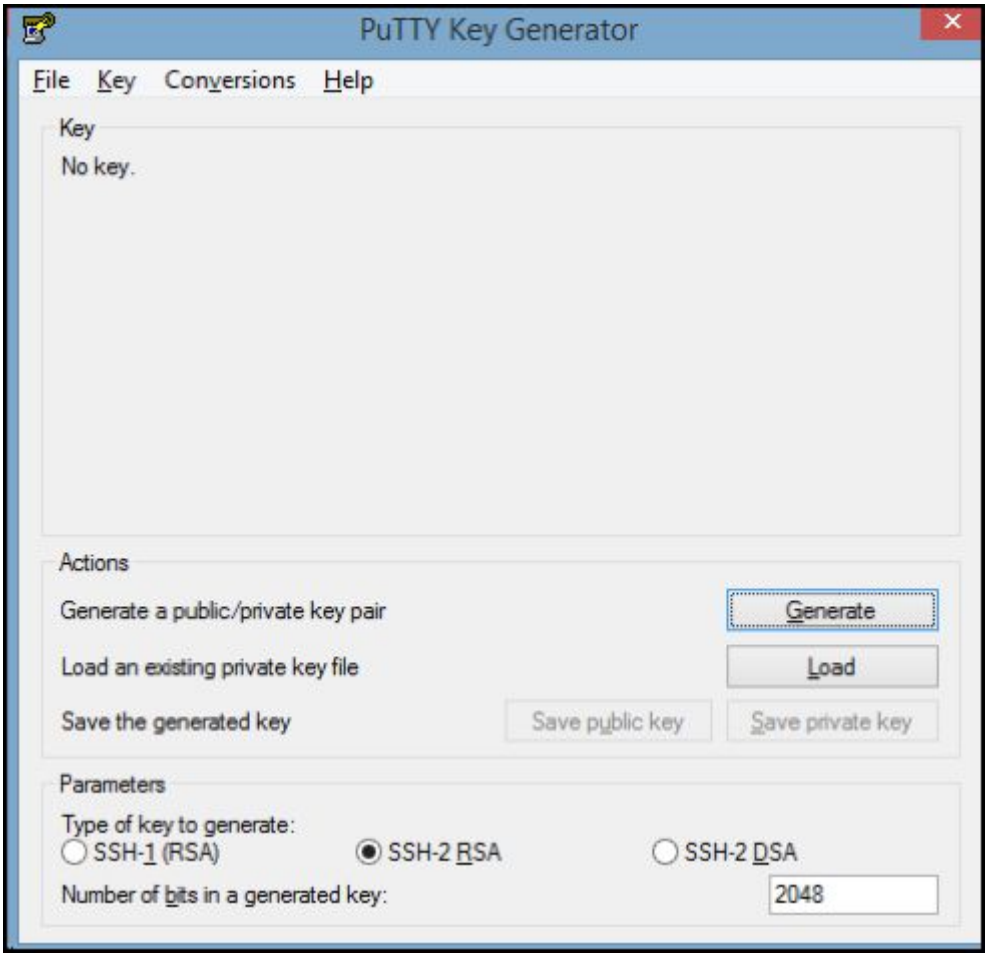
Для доступа по SSH-протоколу предназначен графический клиент PuTTY, загрузить который можно со страницы http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html.

Утилита не имеет установщика и запускается непосредственно из каталога, в котором она расположена. Поле Host Name (or IP address) позволяет задать адрес сервера, Port — порт. Параметры доступа при необходимости могут быть сохранены в список сессий (Saved Sessions), в последующие сеансы они могут быть извлечены кнопкой Load. Установка сеанса осуществляется после нажатия кнопки Open, в результате чего откроется консоль и будет предложено ввести логин пользователя. После проверки логина появится запрос на ввод пароля (при вводе пароля символы не отображаются в консоли). В случае успешной аутентификации откроется консоль удаленного сервера.

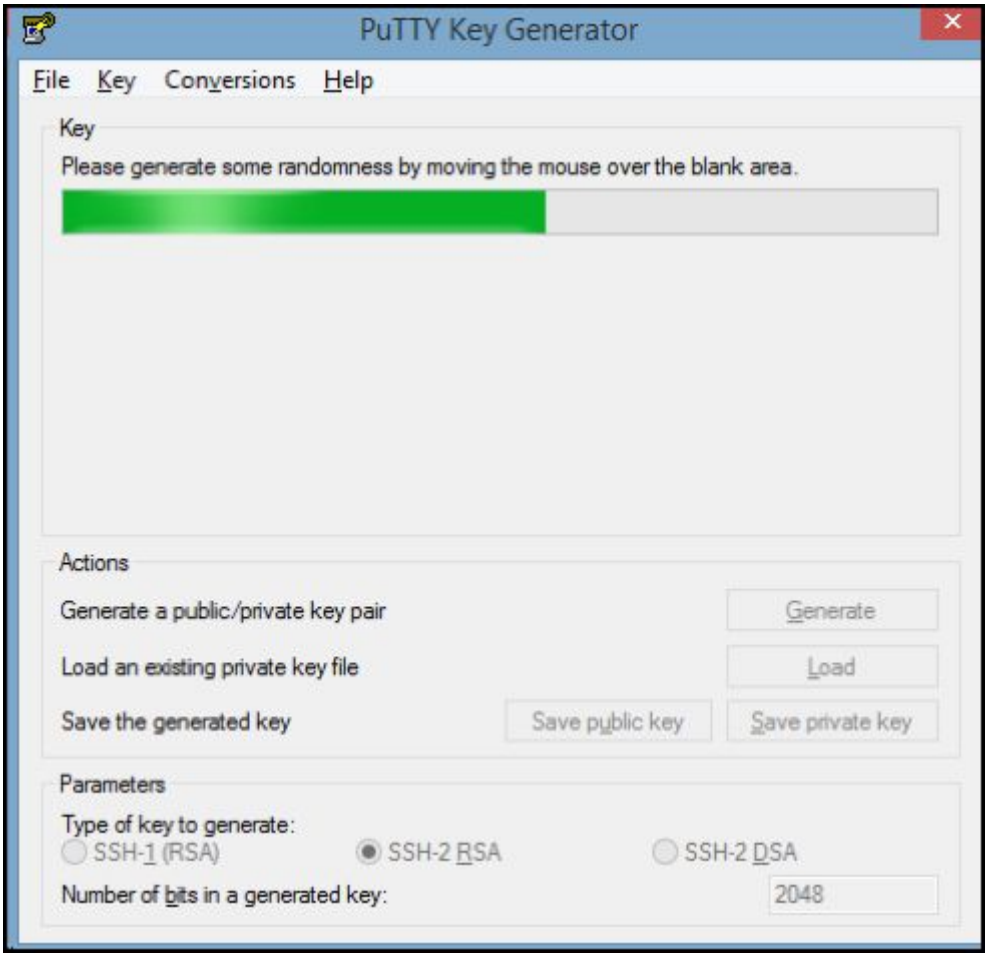
Как и в случае Linux-утилиты ssh, при установке первого соединения с сервером предлагается подтвердить обращение к удаленному серверу при помощи дополнительного диалогового окна.

На странице загрузки http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html есть целый набор различных утилит, позволяющих выполнять те же самые задачи, которые выполняют утилиты в Linux. В том числе, можно организовать доступ по ключу.

Ключи, сгенерированные в UNIX-подобных операционных системах, не подходят для работы с Windows-набором утилит. Вам придется скачать утилиту puttygen.exe и либо сгенерировать новый ключ, либо преобразовать OpenSSH-ключ в формат, который воспринимается Windows-агентом.

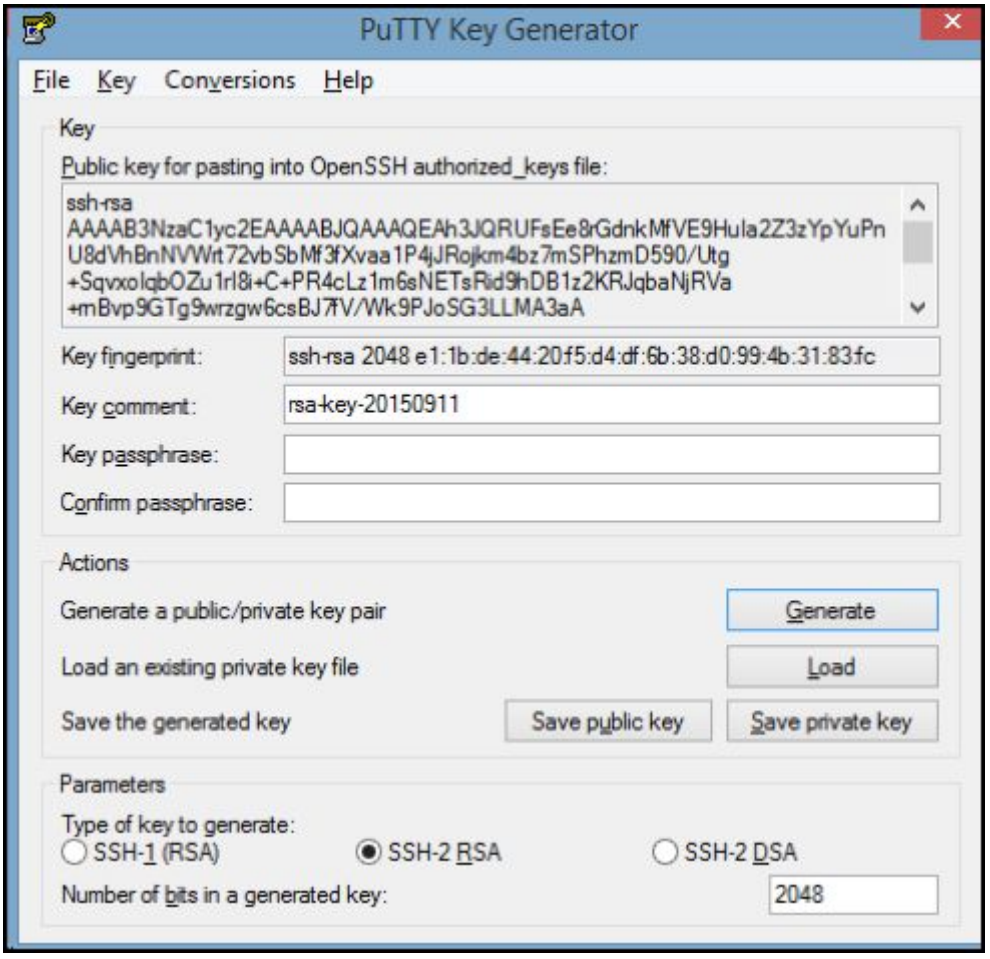



Для создания ключа следует нажать кнопку Generate. Процедура генерации секретного ключа нуждается в последовательности случайных цифр. Для этого утилита предлагает осуществлять случайные действия с мышью и клавиатурой. По мере накопления данных можно наблюдать за полосой прогресса.

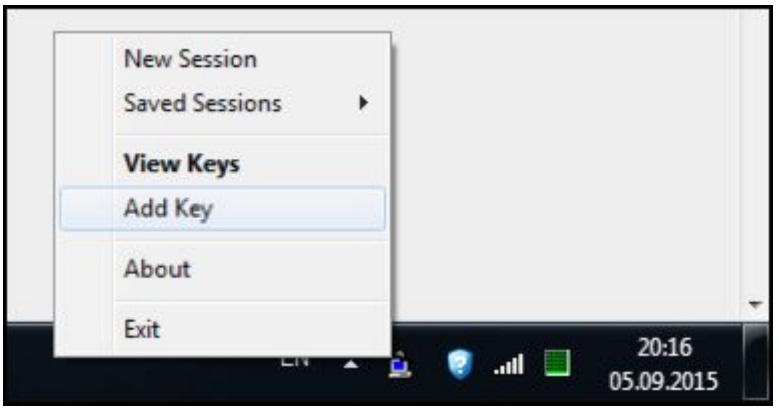



Для преобразования уже существующего ключа следует нажать кнопку Load и загрузить закрытый OpenSSH-ключ, созданный ранее в UNIX-подобной операционной системе. Если закрытый ключ защищен паролем, утилита потребует ввести его.

После генерации или преобразования ключей их следует сохранить: открытый ключ при помощи кнопки Save public key, закрытый — Save private key.




Получив ключи, необходимо запустить агента pageant, который также можно загрузить со страницы http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html. После запуска утилита попадет в область уведомлений (системный трей).




Щелчком правой кнопкой мыши можно вызвать контекстное меню, в котором имеется пункт Add Key, вызывающий диалоговое окно для загрузки закрытого ключа. Если ключ защищен паролем, его придется ввести. Агент будет перехватывать любые SSH-соединения и обеспечивать доступ по загруженному ключу (т. е., без логина и пароля).

Полученные ключи можно использовать в том числе для работы с GitHub.

# Используемые источники

Для подготовки методического пособия мы использовали эти ресурсы:

- Официальная документация Git: **https://git-scm.com/book/ru/v2.**