

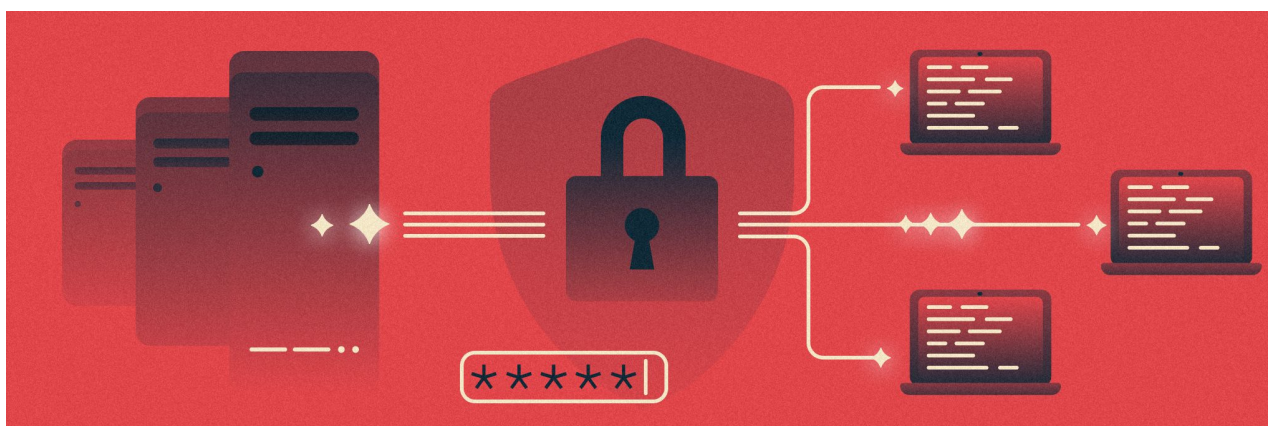
# SSH-туннели: практические примеры использования и важные функции

 [selectel.ru/blog/ssh-tunnels](https://selectel.ru/blog/ssh-tunnels)

28 декабря 2020 г.

Эта инструкция — часть курса «Информационная безопасность на практике».

[Смотреть весь курс](#)



SSH-туннелирование устанавливает защищенный канал связи между локальным рабочим узлом и удаленным сервером. Разбираемся, как можно создавать SSH-туннели, какие полезные для системного администратора функции они имеют и почему важно помнить о безопасности интернет-соединения.

Любые важные данные следует передавать по защищенным каналам информации. Но это не всегда легко реализовать, особенно когда пользователю нужно совершить срочные операции на удаленном сервере, а подключиться к интернету можно только через публичный незащищенный Wi-Fi. В таком случае используют SSH-туннелирование, устанавливающее защищенный канал связи между локальным рабочим узлом и удаленным сервером. С помощью него можно не только передавать или редактировать файлы дистанционно, но и запускать GUI-приложения, делать бэкап, передавать пароли и даже организовывать потоковое вещание.

## Как устроены SSH-туннели

### Создание, условия применения и польза для сисадминов

Ситуация, когда требуется срочный доступ к домашнему компьютеру или внутренней корпоративной сети, знакома каждому. К сожалению, качество и безопасность общественных сетей часто оставляют желать лучшего. Важную информацию по таким сетям лучше не передавать. Кроме того, для организации

доступа часто требуется внешнее ПО (Team Viewer и другие). Для системного администрирования существует способ использования публичных сетей и при этом создания безопасного подключения к необходимым узлам без использования того же VPN. Речь о SSH-туннелировании.

**SSH** (от англ. **Secure Shell** — «безопасная оболочка») — сетевой протокол, который используется для удаленного управления ОС и проксировании TCP-соединений. Выполняет шифрование всего трафика (сюда же относятся пароли и другие важные для корпоративной безопасности данные). SSH-серверы работают с большинством сетевых операционных систем, представленных на рынке.

Подключиться по этому протоколу можно практически к любому серверу. Для организации безопасного соединения можно использовать так называемые SSH-туннели. Но с точки зрения терминологии — это не те туннели, о которых обычно идет речь, когда говорят о системном администрировании. Само наименование, SSH tunnel, сформировалось среди сисадминов для простоты обозначения технологии — SSH Port Forwarding (проброса портов). В ней реализовано сразу несколько возможностей сетевого протокола SSH, а именно — передача TCP-пакетов и трансляция IP-заголовка во время передачи информации при условии существования заранее заданного правила.

Главное отличие SSH tunnels от их аналогов с VPN — передача информации не происходит в любом направлении. Такой канал связи имеет одну точку входа и работает исключительно с TCP-пакетами. Создание SSH-туннелей скорее напоминает проброс портов поверх протокола, нежели туннелирование в чистом виде.

## Как создать SSH-туннель и настроить его параметры

---

Для того, чтобы идентифицировать пользователя, требуется два ключа. Открытый ключ размещается непосредственно на сервере, а закрытый ключ создается и хранится на компьютере пользователя.

### Создать SSH-ключи

Создание ключа и установление соединения с удаленным сервером занимает несколько минут.

### Создание SSH-ключа в Linux (Ubuntu)/MacOS. Пошаговая инструкция:

---

1. Для создания ключа требуется ввести следующую команду:

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

2. После введения команды на экране управляющей консоли появится диалог:

```
Enter file in which to save the key (/home/user/.ssh/id_rsa):
```

3. Чтобы дополнительно защитить соединение, система предложит пользователю придумать и ввести специальное кодовое слово (фразу).

```
Enter passphrase (empty for no passphrase):
```

4. Разумеется, пункт № 3 можно пропустить, нажав **Enter**. То же самое следует сделать, отвечая на следующий вопрос.

5. Завершив создание двух типов ключей (публичного и закрытого), можно переходить к установке связи по SSH. На консоли появится сообщение:

```
Your identification has been saved in /home/user/.ssh/id_rsa.  
Your public key has been saved in /home/user/.ssh/id_rsa.pub.  
The key fingerprint is:  
476:b2:a8:7f:08:b4:c0:af:81:25:7e:21:48:01:0e:98 user@localhost
```

The key's randomart image is:

```
+---[RSA 2048]---+  
|      ..0  o      |  
|      .. * * = .  |  
|      . o O o B .  |  
|      . . + = = o  |  
|      oo S o = . .  |  
|      .. B . = . .  |  
|      . B  o = . .  |  
|      . o.O.. o . .  |  
|      .oooE   o.    |  
+-----[SHA256]-----+
```

6. На следующем этапе в терминале необходимо ввести команду, которая покажет открытый ключ:

```
cat ~/.ssh/id_rsa.pub
```

7. Публичный ключ необходимо ввести в панели управления. Внимательно проверяйте корректность введения ключа.

8. Остается последний этап. Для создания SSH-туннеля к удаленному серверу достаточно выполнить одну команду:

```
ssh root@HOST
```

Параметр HOST — публичный IP-адрес сервера.

Добавим, что введение дополнительных паролей не требуется. А при использовании сервера Selectel можно использовать свой скрипт (bash) для быстрой настройки.

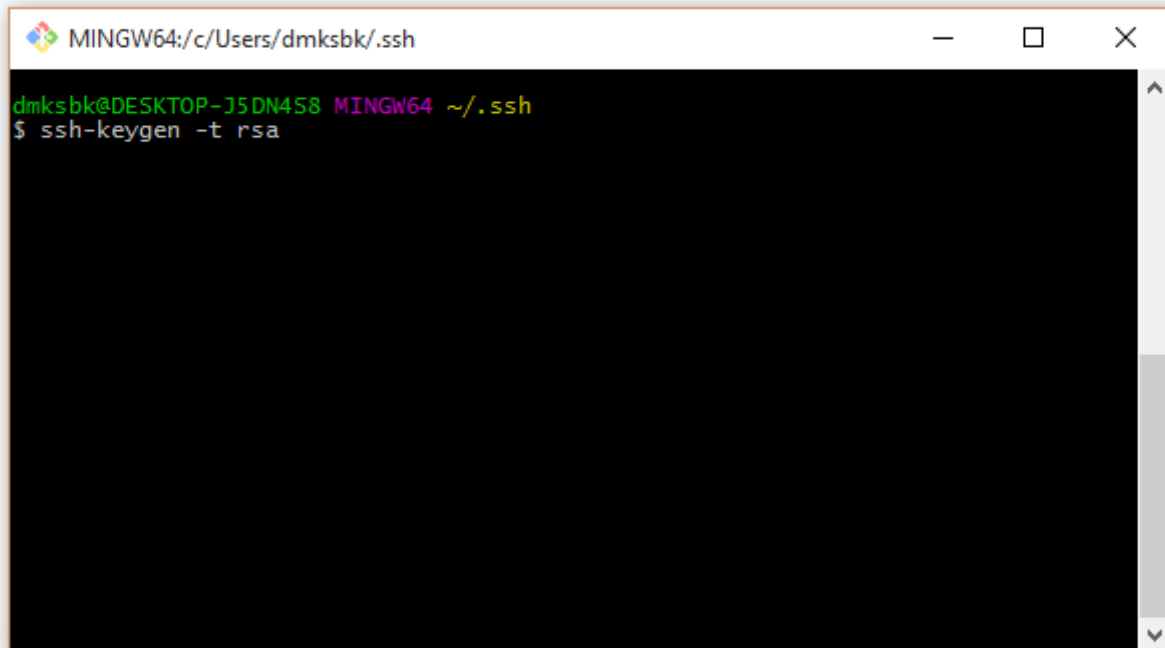
Для удобства пользователей ОС Windows 10 можно создать SSH-туннель в эмуляторе под Windows (MinGW64).

Инструкция для создания SSH-туннеля в Windows 10 выглядит аналогичным образом:

---

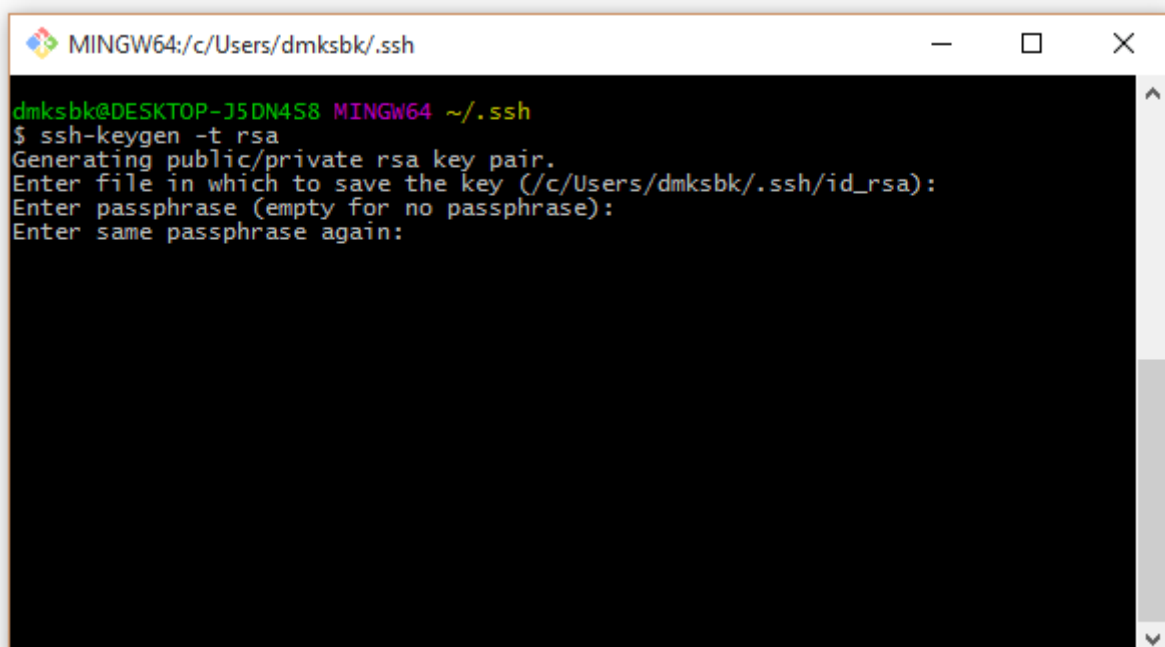
1. Для генерации пары ключей используется команда:

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



A screenshot of a Windows command prompt window titled "MINGW64:/c/Users/dmksbk/.ssh". The prompt shows the user "dmksbk@DESKTOP-J5DN458" in a "MINGW64" environment at the directory "~/.ssh". The command "\$ ssh-keygen -t rsa" has been entered and executed, resulting in a blank line.

2. Желательно защитить приватный ключ паролем (или нажать **Enter**, если он не нужен):



A screenshot of a Windows command prompt window titled "MINGW64:/c/Users/dmksbk/.ssh". The prompt shows the user "dmksbk@DESKTOP-J5DN458" in a "MINGW64" environment at the directory "~/.ssh". The command "\$ ssh-keygen -t rsa" has been entered and executed. The output shows the following prompts: "Generating public/private rsa key pair.", "Enter file in which to save the key (/c/Users/dmksbk/.ssh/id\_rsa):", "Enter passphrase (empty for no passphrase):", and "Enter same passphrase again:". The user has not yet entered a passphrase, so the prompts are still visible.

3. Ключи сгенерированы:

```
MINGW64:/c/Users/dmksbk/.ssh
dmksbk@DESKTOP-J5DN458 MINGW64 ~/.ssh
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/dmksbk/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/dmksbk/.ssh/id_rsa.
Your public key has been saved in /c/Users/dmksbk/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:wPrkKBYkSfPsbqq2ToNCRKQtbgbzC8KU8wKG60na8Us dmksbk@DESKTOP-J5DN458
The key's randomart image is:
+---[RSA 2048]-----+
|. =
|*o= .
|@*oo o
|X*= . .
|oXoo. . S
|B.oo =
|o+=oE o
|+++ =
|*+o o.
+---[SHA256]-----+
dmksbk@DESKTOP-J5DN458 MINGW64 ~/.ssh
```

4. Открытый ключ хранится в файле `~/.ssh/id_rsa.pub`. Приватный ключ находится в файле `id_rsa` и должен храниться в секрете:

```
MINGW64:/c/Users/dmksbk/.ssh
dmksbk@DESKTOP-J5DN458 MINGW64 ~/.ssh
$ ls
id_rsa  id_rsa.pub  known_hosts
dmksbk@DESKTOP-J5DN458 MINGW64 ~/.ssh
$ |
```

5. Чтобы посмотреть содержимое публичного ключа, используется команда:

```
cat ~/.ssh/id_rsa.pub
```

```
MINGW64:/c/Users/dmksbk/.ssh
dmksbk@DESKTOP-J5DN4S8 MINGW64 ~/.ssh
$ ls
id_rsa  id_rsa.pub  known_hosts

dmksbk@DESKTOP-J5DN4S8 MINGW64 ~/.ssh
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDA8e1Po2JVXb8RXBcNt7iJxGkDlj1PmjObdNvFcG6lboCodop3vV1x1lI1ed1GF7tycNbxP5BmmPe9B6e5an1U85LJ1RtU8T7KeG3PdxUKKX/+CJZKkUqCopsSmQtMN9JcwihqUB6f8StggKvDhrW5NFy62CArGhjy5ie+llfm9ZfPiTJF1nwwVr1YTbt5PTX7kxa+53EuBpOuF+WX0gJI0ui+4BZTVUihp6ev5uRiZOJH2Akbo1bx96eNDYR3VLA5tPTui1WrXAHlWNp9L9BxNgY+YJkYwMMeoI43rZ5mQIcKinSvhNvyj/pgd+q9w5R9jNNzPm3GhcoiG9Ktd0sUr dmksbk@DESKTOP-J5DN4S8

dmksbk@DESKTOP-J5DN4S8 MINGW64 ~/.ssh
$
```

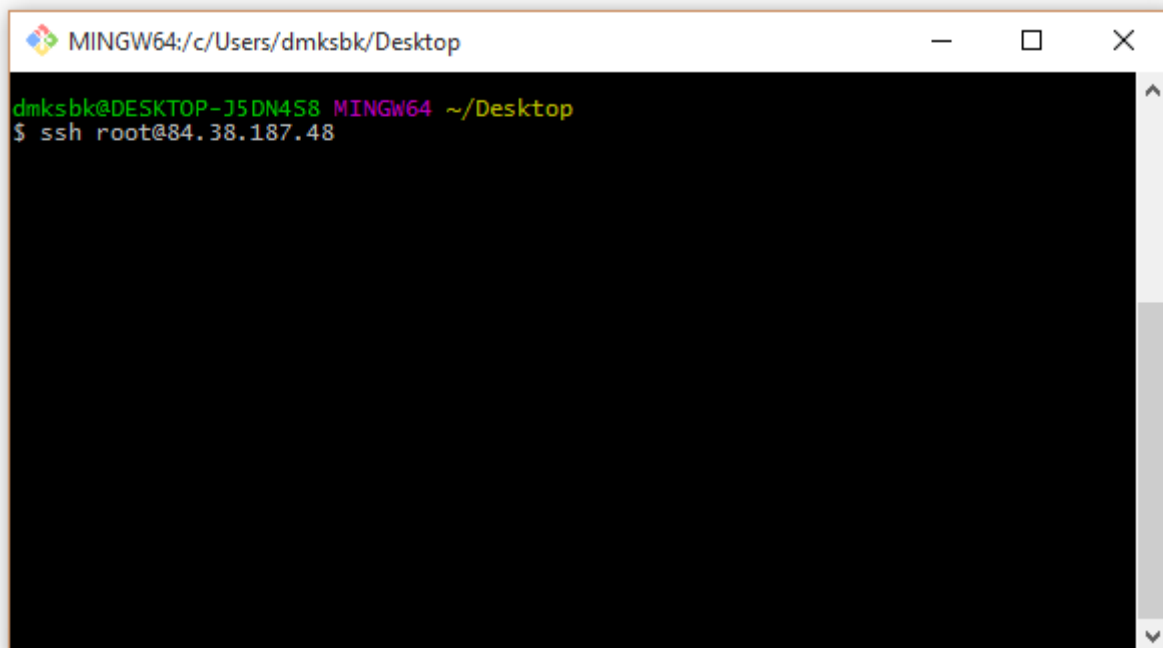
6. Этот ключ (всю строку, начинающуюся с **ssh-rsa**) необходимо вставить в панели управления Selectel, в поле **Операционная система / SSH-ключ выбранного сервера**. При установке ОС он будет скопирован в файл **.ssh/authorized\_keys**. При смене ключа в панели управления Selectel текущий образ ОС будет пересоздан, а данные на сервере — утеряны.

SSH-ключ

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDA8e1Po2JVXb8RXBcNt7iJxGkDlj1PmjObdNvFcG6lboCodop3vV1x1lI1ed1GF7tycNbxP5BmmPe9B6e5an1U85LJ1RtU8T7KeG3PdxUKKX/+CJZKkUqCopsSmQtMN9JcwihqUB6f8StggKvDhrW5NFy62CArGhjy5ie+llfm9ZfPiTJF1nwwVr1YTbt5PTX7kxa+53EuBpOuF+WX0gJI0ui+4BZTVUihp6ev5uRiZOJH2Akbo1bx96eNDYR3VLA5tPTui1WrXAHlWNp9L9BxNgY+YJkYwMMeoI43rZ5mQIcKinSvhNvyj/pgd+q9w5R9jNNzPm3GhcoiG9Ktd0sUr
dmksbk@DESKTOP-J5DN4S8
```

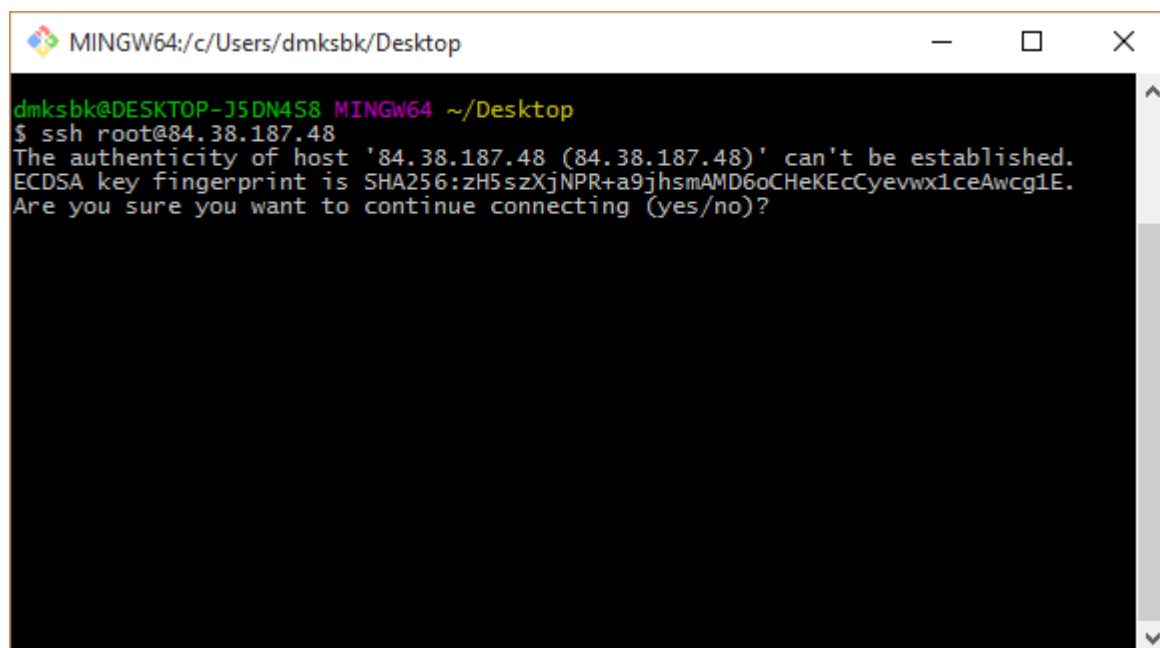
7. Для подключения к серверу по SSH используется команда:

```
ssh root@84.38.187.48 (IP-адрес сервера будет другим).
```



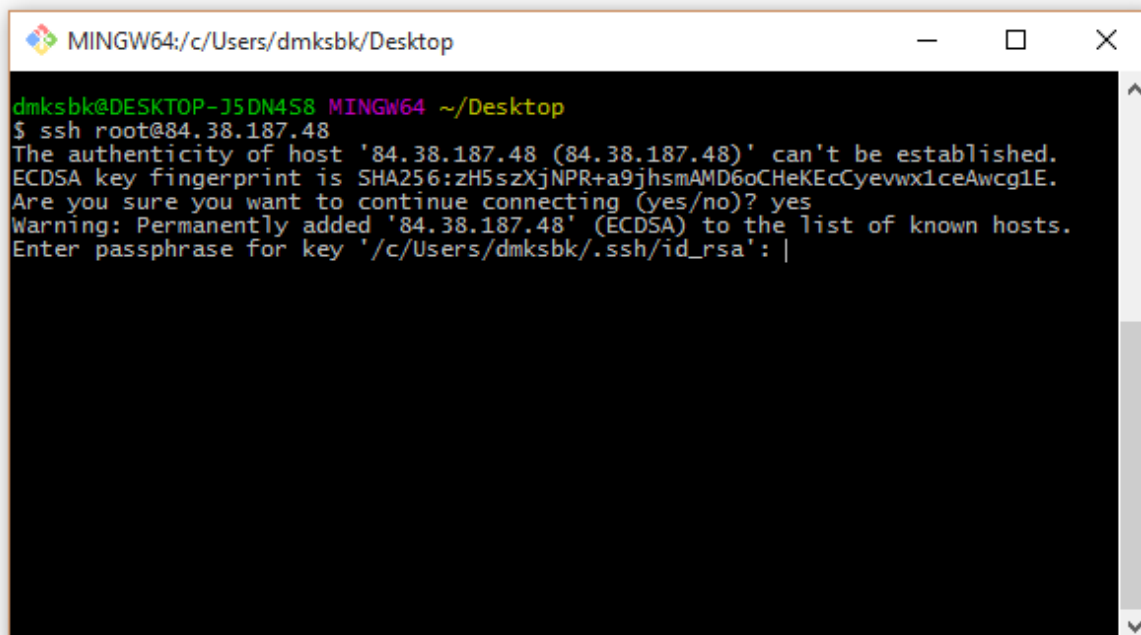
```
MINGW64:/c/Users/dmksbk/Desktop
dmksbk@DESKTOP-J5DN458 MINGW64 ~/Desktop
$ ssh root@84.38.187.48
```

8. При первом подключении утилита **SSH** сообщает о неизвестном хосте и спрашивает, уверены ли вы, что доверяете ему. Если ответ положительный (следует ответить **yes**), IP-адрес и открытый сертификат этого сервера будут добавлены в файл `~/.ssh/known_hosts` на локальной машине:



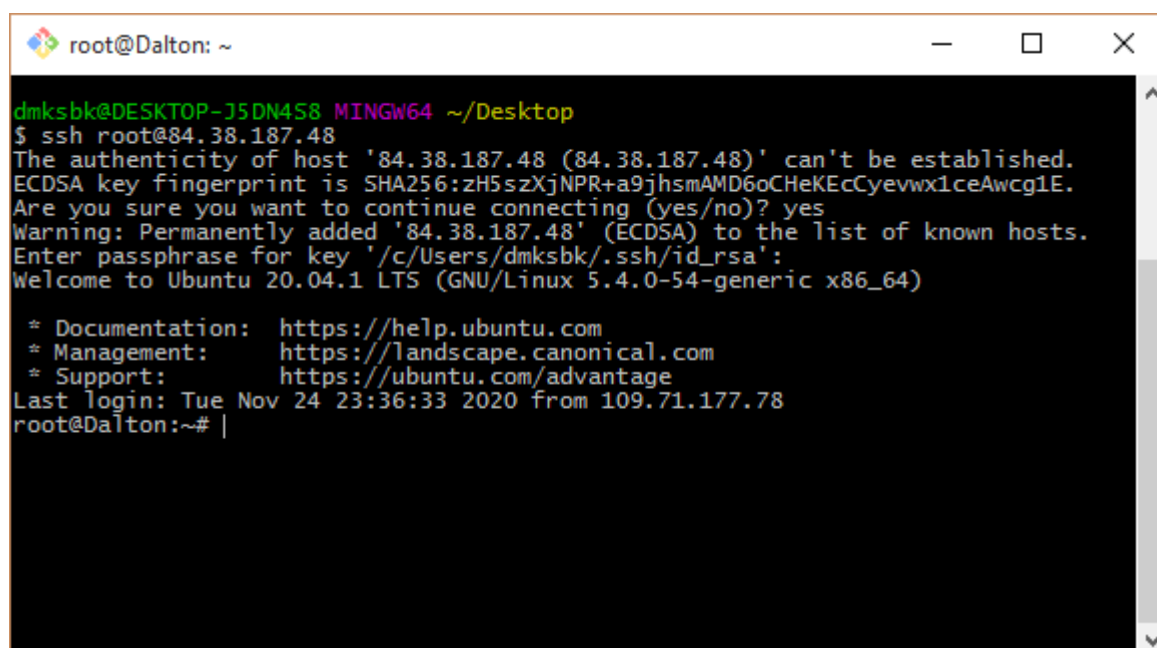
```
MINGW64:/c/Users/dmksbk/Desktop
dmksbk@DESKTOP-J5DN458 MINGW64 ~/Desktop
$ ssh root@84.38.187.48
The authenticity of host '84.38.187.48 (84.38.187.48)' can't be established.
ECDSA key fingerprint is SHA256:zH5szXjNPR+a9jhsmAMD6oCHeKEcYevwx1ceAwcg1E.
Are you sure you want to continue connecting (yes/no)?
```

9. Если ранее при генерации пары ключей был задан пароль, утилита **SSH** попросит ввести его:



```
MINGW64:/c/Users/dmksbk/Desktop
dmksbk@DESKTOP-J5DN4S8 MINGW64 ~/Desktop
$ ssh root@84.38.187.48
The authenticity of host '84.38.187.48 (84.38.187.48)' can't be established.
ECDSA key fingerprint is SHA256:zH5szXjNPR+a9jhsmAMD6oCHeKEcCyevwx1ceAwcg1E.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '84.38.187.48' (ECDSA) to the list of known hosts.
Enter passphrase for key '/c/Users/dmksbk/.ssh/id_rsa': |
```

10. После ввода пароля и нажатия **Enter** устанавливается зашифрованное соединение с сервером:



```
root@Dalton: ~
dmksbk@DESKTOP-J5DN4S8 MINGW64 ~/Desktop
$ ssh root@84.38.187.48
The authenticity of host '84.38.187.48 (84.38.187.48)' can't be established.
ECDSA key fingerprint is SHA256:zH5szXjNPR+a9jhsmAMD6oCHeKEcCyevwx1ceAwcg1E.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '84.38.187.48' (ECDSA) to the list of known hosts.
Enter passphrase for key '/c/Users/dmksbk/.ssh/id_rsa':
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
Last login: Tue Nov 24 23:36:33 2020 from 109.71.177.78
root@Dalton:~# |
```

Еще один вариант, как можно использовать этот тип соединения, — создание канала связи по протоколу RDP. Этот вариант подходит, когда пользователь не может подключиться к узлу из-за отсутствия доступа к нему из сети. Зато у него есть доступ к маршрутизатору. Этого достаточно, чтобы использовать данный тип туннелирования.



Параметры канала связи устанавливает инициатор подключения. На втором конце канала связи всегда расположен целевой сервер. Так, клиентом может оказаться VPS-сервер, а точкой выхода — компьютер сисадмина. Точка входа может быть создана с любой стороны туннеля: именно через нее принимаются подключения. Обратная сторона туннеля — точка выхода — способна лишь на маршрутизацию пакетов информации с учетом установленных правил. Отдельная опция — соединение с удаленной машиной, на которой запущен модуль Docker (SSH Docker). Подробнее об этом мы расскажем в следующих публикациях.

Аналогичным образом проходит процесс установки соединения с определенным сервером. В качестве примера возьмем целевой сервер с адресом 192.168.0.105. При этом для пользователя доступен только маршрутизатор (192.168.0.1). В качестве точки входа прописывается 127.0.0.1:3389. Кроме того, задается правило трансляции, определяющее узел, принимающий данные на выходе из SSH-туннеля. В нашем случае в качестве правила указывается 192.168.0.105:3389. Проследите, чтобы указанный адрес действительно существовал, иначе вам не удастся подключиться к целевому серверу.

После того, как вы зададите исходные данные, появится TCP-сокеты, который создает служба SSH. Этот локальный сокет следит за тем, когда начнутся подключения на порт 3389. Точкой назначения прописывается localhost (127.0.0.1). Отметим, что RDP-клиент не владеет информацией о настоящем назначении получателя пакета. Клиент только открывает динамический порт, чтобы отправить пакет данных, у которого задан адрес назначения (точка входа) и источника (127.0.0.1:61256).

Пакет с информацией следует от входной точки SSH tunnel на его противоположный конец. Адрес трансляции поменяется. Теперь там значится 192.168.0.105:3389. На следующем этапе SSH-сервер просто поставит свой адрес вместо адреса источника. Вся информация, которая передается внутри канала, будет отправлена поверх протокола. А RDP-клиент оперирует исключительно локальным сокетом. Важно, что вся информация внутри туннеля защищена — она шифруется. Но соединение между SSH- и RDP-серверами остается обычным. Разумеется, этот фактор стоит учитывать, если пользователь работает с небезопасными протоколами.

## **SSH Proxy: как организовать доступ к любой системе (обращение к проксируемому серверу)**

---

С помощью SSH-туннеля можно воспользоваться домашней (или корпоративной) сетью, даже используя ненадежный бесплатный Wi-Fi. Для этого пользователю потребуется запустить SSH-прокси, а затем по аналогичному принципу создать туннель в целевую сеть (например, можно подключиться к домашней сети).

Важно, чтобы приложения, которые потребуются инициатору подключения для работы, поддерживали SOCKS-прокси. С помощью туннеля сетевые службы станут доступны для удаленной работы. Другой вариант: пользователь сможет выйти в интернет, используя домашнее подключение, но сидя при этом в другом конце мира и используя публичную сеть. Вся передаваемая через туннель информация будет зашифрована.

Это не единственный способ практического применения SSH-туннелирования. Один из самых популярных кейсов — SSH Proxy — открывает доступ к желаемой системе, если она доступна для удаленного сервера. При этом для туннелирования через прокси-сервер потребуется ввести всего одну команду:

```
localhost:~$ ssh -D 8888 user@remoteserver
```

```
localhost:~$ netstat -pan | grep 8888
tcp        0      0 127.0.0.1:8888      0.0.0.0:*           LISTEN
23880/ssh
```

Команда выше показывает, что пользователь запускает SOCKS-прокси. Портом для работы назначается 8888. Кроме того, необходимо проверить, активен ли этот TCP-порт (используется режим прослушивания). Служба работает исключительно на localhost (127.0.0.1). Если немного видоизменить команду, то можно прослушивать все интерфейсы (в том числе ethernet или Wi-Fi). Это позволяет приложениям подключаться к прокси-серверу через SSH-прокси:

```
localhost:~$ ssh -D 0.0.0.0:8888 user@remoteserver
```

Чтобы браузер работал корректно, необходимы настройки. Например, чтобы запустить Chrome с активированным SOCKS-прокси, потребуется команда:

```
localhost:~$ google-chrome --proxy-server="socks5://192.168.1.10:8888"
```

Она создает SOCKS-прокси, а также иницирует туннелирование DNS-запросов. Утилита tcpdump проверяет, видны DNS-запросы или нет.

Кроме браузеров, многие современные приложения работают с SOCKS-прокси: достаточно изменить их параметры, чтобы активировать прокси-сервер. Отдельно применяются инструменты, которые помогают приложениям использовать эту важную функцию — к примеру, proxchains позволяет запустить через SOCKS-прокси Microsoft RDP.

## Зачем нужны динамические SSH tunnels

---

Динамический SSH tunnel отличается от рассмотренных выше типов соединений. Он открывает локальный TCP-сокет для его использования в качестве SOCKS4/SOCKS5 прокси. Обычно его применяют, когда требуется VPN, а его развертывание невозможно. Такой тип SSH-туннеля также отвечает необходимым требованиям безопасности.

Кроме того, создание динамического туннеля удобно, когда пользователю необходимо выйти в интернет однократно:

```
ssh -D 1080 user@vps.example.com
```

SOCKS-прокси работает через порт 1080.

Помимо этого, динамический туннель не предусматривает открытие дополнительных портов во внешнюю сеть. Весь трафик будет проходить исключительно через SSH-соединение, скрывая тем самым характер интернет-деятельности пользователя.

## Команды и практическое применения функций SSH-туннелей

---

Рассмотрим несколько практических примеров применения SSH-туннелирования.

### Проброс (перенаправление) портов — Port Forwarding

---

Чтобы осуществить проброс портов, потребуется создание SSH tunnel. Для этого в локальной системе открывается порт, а для создания канала связи пользователь должен выбрать порт на другом конце туннеля:

```
localhost:~$ ssh -L 9999:127.0.0.1:80 user@remoteserver
```

Команда устанавливает прослушку на порт 9999. Порт 80 используется для проброса. Для Port Forwarding также используются прокси-серверы или TCP-службы.

### Автоматизация копирования публичного (открытого) ключа

---

Традиционно пользователю приходится копировать открытые ключи вручную. SSH-протокол существенно ускоряет этот процесс:

```
localhost:~$ ssh-copy-id user@remoteserver
```

Эта команда выполняет копирование публичного ключа по умолчанию или из директории `~/.ssh/id_rsa.pub` в `~/.ssh/authorized_keys` (она находится на удаленном сервере).

### Обратный SSH-туннель (SSH reverse tunnel)

---

Очевидно, что SSH tunnel можно создать и в обратном направлении. Достаточно подключить прослушивающий порт к другому локальному порту:

```
localhost:~$ ssh -v -R 0.0.0.0:1999:127.0.0.1:902 192.168.1.100 user@remoteserver
```

Данный туннель работает следующий образом: от порта 1999 к remoteserver, а уже после этого обращается к порту 902 на локальном клиенте.

### Удаленное выполнение команд с помощью SSH

---

SSH-команда позволяет создать интерфейс для работы с командами на удаленном хосте. Они прописываются в качестве последнего параметра:

```
localhost:~$ ssh remoteserver "cat /var/log/nginx/access.log" | grep badstuff.php
```

После скачивания лога грег можно запустить на удаленной стороне.

## Копирование, функция **rsync**-копирования (**rsync** через **SSH**)

---

Чтобы создать дубликат папки на удаленном сервере, достаточно сначала сжать папку с помощью **bzip2**, а после извлечь поток **bzip2** на другой стороне. Команда для этой операции следующая:

```
localhost:~$ tar -cvj /datafolder | ssh remoteserver "tar -xj -C /datafolder"
```

Для регулярных бэкапов важной информации используется **rsync**:

```
localhost:~$ rsync -az /home/testuser/data proglibserver:backup/
```

Функция **rsync** позволяет копировать отличия, сравнивая информацию в разных временных точках. Таким образом, можно восстановить важные данные в случае неудачного сеанса передачи.

## GUI-приложения: удаленный запуск через **SSH**

---

SSH-туннель поддерживает функцию, позволяющую запускать GUI-приложения удаленно:

```
localhost:~$ ssh -X remoteserver vmware
```

Отметим: несмотря на то, что GUI выполняется на удаленном сервере, окно отображается на локальном рабочем столе. В примере для работы запущена консоль виртуальной машины VMware Workstation. Но для успешной реализации команды требуются пакеты **X11** и **YES** для **X11Forwarding** в файле **sshd\_config**.

## Редактирование текстовых файлов

---

Метод редактирования файлов одной командой предусматривает создание файла в **/tmp**, а затем копирование в заданную директорию:

```
localhost:~$ vim scp://user@remoteserver/etc/hosts
```

## Прыжки по хостам

---

Туннелирование предусматривает переход через несколько хостов, если пользователь сталкивается с сегментацией сети:

```
localhost:~$ ssh -J host1,host2,host3 user@host4.internal
```

Параметр `-J` использует переадресацию для установления сеанса с каждым следующим хостом в цепочке. При этом рабочий сеанс полностью зашифрован — от `localhost` и до `host4`.

## Фильтрация трафика с помощью iptables

---

Утилита `iptables` позволяет устанавливать правила для блокировки или разрешения прохождения трафика: `INPUT`, `FORWARD` и `OUTPUT`. В случае если пользователь не задаст никаких правил, этот межсетевой экран будет выполнять заданную по умолчанию фильтрацию. Для установления связи с целевым сервером утилита сравнит IP-адрес инициатора подключения с тем списком, что есть в правилах `INPUT`, а затем либо даст доступ к серверу, либо запретит его.

Однако работа с `iptables` требует определенной осторожности. Причина проста: если неверно задать правило фильтрации, удаленный доступ может оказаться просто невозможным до тех пор, пока пользователь не снимет ограничения, получив физический доступ к компьютеру.

## Реализация SSH tunnels в Windows

---

Возможности туннелирования доступны и для тех, кто использует Windows 10: под эту ОС существует ряд SSH-клиентов. Один из самых применяемых — PuTTY. Запуск сервера под Windows — более сложная задача, которая требует от пользователя специализированной квалификации.

PuTTY настраивается достаточно просто: для этого во вкладке `Connection` открываем `SSH Tunnels`, где необходимо прописать базовые настройки соединения — `Source port` (точка входа), `Destination` (назначение); радиопереключатели `Local` — `Remote` — `Dynamic` определяют тип будущего канала связи. Адрес целевого узла указывается в другом разделе — `Session`.

Открытия дополнительных портов или развертывания VPN не требуется.

## Безопасны ли SSH-туннели

---

Как мы уже отмечали, одна из их задач — создание безопасного соединения с удаленной машиной. Однако не стоит забывать о «побочных эффектах» таких туннелей. Кроме соединения с удаленным сервером, пользователь получает консоль, которая открывается на сервере.

Если пользователь забудет о своем подключении, то может выполнить те команды, которые изначально были предназначены к выполнению на локальном узле. Чтобы избежать подобной ошибки (особенно если пользователь имеет права суперпользователя), при запуске канала связи следует указывать параметр `-N`:

```
ssh -N -L -g 3389:192.168.0.105:3389 user@rt.example.com
```

Кроме того, не стоит использовать при подключении к удаленной машине аккаунт суперпользователя, для этих целей подойдет учетная запись с обычными правами. И не забывайте про то, что по завершении работы требуется удалить SSH-туннель.

8 минут

20 шагов для повышения уровня информационной безопасности

Предыдущий материал

15 минут

Что такое брутфорс: какие у него цели и кому это нужно

Следующий материал

## **Зарегистрируйтесь в панели управления**

---

И уже через пару минут сможете арендовать сервер, развернуть базы данных или обеспечить быструю доставку контента.