Использование ssh-keygen для генерации ключей SSH: подробное руководство

Взято отсюда: https://www.securitylab.ru/analytics/562583.php

Подробное руководство по генерации и использованию SSH-ключей с помощью ssh-keygen. Объяснение принципов работы, инструкции для Linux, macOS и Windows.

Помните времена, когда для входа на сервер нужно было каждый раз вводить пароль? Как будто недостаточно головной боли с запоминанием паролей к соцсетям, почте и банковским приложениям. Хорошо, что умные люди придумали SSH-ключи — систему, где вместо ежедневного «угадай пароль» вы один раз настраиваете пару ключей и забываете про эту проблему.

SSH-ключи стали золотым стандартом для доступа к серверам, Git-репозиториям и облачной инфраструктуре. Принцип простой: есть два ключа — открытый и закрытый. Закрытый остается у вас и охраняется как зеница ока, а открытый можно спокойно раздавать направо и налево. Попытка подобрать ключ методом перебора займет у злоумышленника примерно столько же времени, сколько нужно на пересчет всех песчинок на планете.

В центре этой магии — утилита ssh-keygen, которая входит в комплект OpenSSH. Она генерирует ключи, меняет пароли, показывает отпечатки и вообще делает всю черновую работу. Ниже — практическое руководство без воды: от выбора алгоритма до интеграции с агентом и настройки аппаратных токенов безопасности.

Почему ключи надежнее паролей (и почему об этом вообще стоит думать)

Асимметричная криптография работает примерно как почтовый ящик с хитрым замком. У вас есть два ключа: один открывает ящик для складывания писем (это открытый ключ), а второй — для их извлечения (закрытый ключ). Любой может положить письмо в ящик, но достать его может только владелец второго ключа.

В случае с SSH закрытый ключ никогда не покидает ваш компьютер. Сервер хранит копию вашего открытого ключа в файле ~/.ssh/authorized_keys, и когда вы подключаетесь, происходит математическая «рукопожатие»: сервер проверяет, что у вас действительно есть закрытая половина ключа. Даже если кто-то перехватит весь трафик, без закрытого ключа он ничего не сможет сделать.

Преимущества такой схемы очевидны: никаких паролей в сети, никаких «admin123» и «qwerty», никаких попыток подобрать пароль перебором. Если закрытый ключ надежно защищен фразой-паролем и правильными правами доступа к файлам, это практически неприступная крепость.

- Открытый ключ: безобидная текстовая строка в файле *.pub, которую можно публиковать

где угодно
- Закрытый ключ: ваша тайная половина, которую нужно беречь как код от банковской карты
- Фраза-пароль: дополнительная защита на случай, если кто-то все-таки доберется до файла

Выбираем алгоритм: почему ED25519 стал новым королем

В 2025 году OpenSSH официально поменял алгоритм по умолчанию с RSA на ED25519. И это не просто дань моде — ED25519 действительно лучше по всем параметрам: компактнее, быстрее, криптографически стойкий и поддерживается практически везде, где есть современный SSH.

RSA все еще работает и широко поддерживается, но требует ключей длиной минимум 3072 бита для приемлемой безопасности. ED25519 при той же стойкости умещается в гораздо меньший размер. ECDSA тоже неплох, но у него есть свои тонкости с источниками случайности, поэтому в новых установках обычно выбирают ED25519.

Отдельная категория — аппаратные ключи на FIDO2-токенах (YubiKey и подобные). Они хранят секретный материал внутри защищенного устройства и могут требовать физического касания для подтверждения входа. Это резко усложняет жизнь злоумышленникам, даже если они получили доступ к вашему компьютеру.

- Для новых проектов: ed25519 — современно, быстро, надежно
- Для совместимости со стариной: rsa с длиной от 3072 бит
- Для параноиков: ed25519-sk или ecdsa-sk с FIDO2-токеном
- Документация: мануал ssh-keygen, настройки ssh_config

Где живут ключи и почему SSH такой привередливый к правам доступа

По умолчанию ключи лежат в папке ~/.ssh в домашнем каталоге. Там будут файлы типа id_ed25519 (закрытый ключ) и id_ed25519.pub (открытый). На одной машине может быть несколько ключей для разных задач — ничего страшного, главное не путать их назначение и вести учет.

SSH очень придирчив к правам доступа на файлы, и это правильно. Если ваш секретный ключ может читать кто угодно, значит, он уже не секретный. Поэтому при неправильных правах SSH откажется работать с хмурым сообщением «permissions are too open».

- Папка ~/.ssh: chmod 700 ~/.ssh (только владелец может читать и писать)
- Закрытые ключи: chmod 600 ~/.ssh/id_* (только владелец может читать)
- Открытые ключи: обычно 644 хватает (но можно и строже)
- На сервере: ~/.ssh/authorized_keys — 600, папка ~/.ssh — 700

Быстрый старт: создаем современный ключ за пять минут

Этот рецепт покроет 90% случаев. Генерируем ED25519-ключ с понятным комментарием и надежной защитой. Флаг -a добавляет «соль в рану» для тех, кто попытается подобрать фразу-пароль методом перебора.

```
ssh-keygen -t ed25519 -a 100 -C "Petr Ivanov work noutbook 2025" -f
~/.ssh/id_ed25519
```

Утилита спросит, где сохранить ключ и какую фразу-пароль установить. Если вы работаете на личном защищенном компьютере, можете оставить фразу пустой, но в корпоративной среде это моветон. Потерять ноутбук без защиты ключа — это почти как оставить в нем записку с паролями ко всем серверам.

- -t ed25519 — выбираем алгоритм (самый современный на сегодня)
- -a 100 — усложняем подбор фразы-пароля (100-200 — золотая середина)
- -C — комментарий для опознания ключа через пару лет
- -f — точно указываем путь к файлу

Для особых случаев: RSA, кастомные параметры и тихий режим

Иногда приходится работать с древними системами, где ED25519 еще не завезли, или корпоративная политика безопасности требует конкретных параметров. Тогда можно создать RSA-ключ с увеличенной длиной и современным форматом хранения:

```
ssh-keygen -t rsa -b 4096 -o -a 150 -C "RSA for old server" -f
~/.ssh/id_rsa_legacy -q
```

Флаг -o сохраняет ключ в современном формате OpenSSH (защищенном через bcrypt), а -q включает «тихий режим» — полезно в скриптах, где не нужна болтовня в консоль.

- -b 4096 — длина RSA-ключа (чем больше, тем безопаснее, но медленнее)
- -o — современный формат вместо старого PEM
- -q — меньше текста в выводе

Аппаратные ключи: когда паранойя оправдана

Если у вас есть FIDO2-токен (YubiKey, SoloKey или аналогичный), стоит попробовать аппаратные ключи SSH. Секретный материал живет внутри токена и никогда его не покидает, а для входа на сервер может потребоваться физическое касание устройства. Даже если злоумышленник полностью скомпрометирует ваш компьютер, без токена он не сможет подключиться к серверам.

```
ssh-keygen -t ed25519-sk -C "Hardware key for Production" -f
~/.ssh/id_ed25519_sk
```

Полезные опции для FIDO-ключей:

- -O resident — сохранить открытый ключ прямо на токене (удобно при смене компьютера)
- -O verify-required — требовать PIN или биометрию перед использованием
- -O application=ssh: — задать идентификатор приложения
- Подробности: мануал ssh-keygen, история FIDO2 в OpenSSH: заметки к релизу 8.2

Добавляем ключ на сервер: автоматически или руками

Самый простой способ — утилита ssh-copy-id. Она сама скопирует открытый ключ на сервер и поставит правильные права доступа:

```
ssh-copy-id -i ~/.ssh/id_ed25519.pub user@server.example.com
```

Если ssh-copy-id недоступна (например, на некоторых версиях macOS или в минимальных дистрибутивах), можно сделать то же самое вручную:

```
cat ~/.ssh/id_ed25519.pub | ssh user@server.example.com \
'mkdir -p ~/.ssh && cat >> ~/.ssh/authorized_keys && chmod 600
~/.ssh/authorized_keys && chmod 700 ~/.ssh'
```

- Проверьте права на сервере: ~/.ssh должна быть 700, authorized_keys — 600
- Убедитесь, что в /etc/ssh/sshd_config включен PubkeyAuthentication yes
- Документация: ssh-copy-id manual

SSH-агент: один раз ввел пароль — день работаешь спокойно

Агент SSH — это такой добрый демон, который запоминает разблокированные ключи и подставляет их при необходимости. Один раз утром ввели фразу-пароль, и весь день подключаетесь к серверам без дополнительных вопросов. Это экономит кучу времени и снимает соблазн создавать ключи вообще без защиты.

На большинстве Linux-дистрибутивов и в macOS агент запускается автоматически. Если нет — можно стартануть вручную:

```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_ed25519
```

В macOS есть фишка с интеграцией в «Связку ключей» — система сама достанет фразу-пароль при входе в аккаунт:

```
# один раз добавляем ключ в Связку
ssh-add --apple-use-keychain ~/.ssh/id_ed25519

# прописываем в ~/.ssh/config
Host *
```

```
  UseKeychain yes
  AddKeysToAgent yes
  IdentityFile ~/.ssh/id_ed25519
```

В Windows тоже есть полноценный SSH-агент как системная служба. Подробности: документация Microsoft.

Файл ~/.ssh/config: как превратить SSH в удобный инструмент

Конфигурационный файл — это способ навсегда избавиться от длинных команд типа ssh -i ~/.ssh/special_key -p 2222 deploy@long-server-name.company.com. Вместо этого прописываете псевдонимы и подключаетесь простой командой ssh prod.

Пример конфигурации для трех окружений:

```
Host prod
  HostName prod.example.com
  User deploy
  Port 22
  IdentityFile ~/.ssh/id_ed25519
  IdentitiesOnly yes

Host staging
  HostName 203.0.113.25
  User deploy
  IdentityFile ~/.ssh/id_ed25519
  ProxyJump bastion.example.com

Host vps
  HostName vps.provider.net
  User root
  IdentityFile ~/.ssh/id_ed25519_sk
```

- IdentitiesOnly yes запрещает SSH перебирать все ключи подряд
- ProxyJump настраивает подключение через промежуточный сервер
- Все опции: руководство по ssh_config

Комментарии и именование: порядок в файлах — порядок в голове

Комментарий в открытом ключе — это сообщение для будущего себя. Через год вы точно не вспомните, какой ключ для чего создавали, поэтому лучше сразу писать что-то вменяемое: «рабочий ноутбук 2025», «ключ для GitHub», «сервер базы данных» и так далее.

То же самое касается имен файлов — не бойтесь отходить от стандартных id_rsa и называть ключи по их назначению:

- Осмысленные комментарии: -C «Anna Petrova MacBook Pro 2025»
- Говорящие имена: ~/.ssh/id_ed25519_github, ~/.ssh/id_ed25519_work
- В конфигурации явно указывайте IdentityFile для каждого хоста

Смена паролей и восстановление потерянных файлов

Если нужно сменить фразу-пароль у существующего ключа, это делается одной командой:

```
ssh-keygen -p -f ~/.ssh/id_ed25519
```

Утилита попросит старую и новую фразы. Если потерялся файл *.pub, но закрытый ключ на месте, открытую часть можно восстановить:

```
ssh-keygen -y -f ~/.ssh/id_ed25519 > ~/.ssh/id_ed25519.pub
```

- Посмотреть «отпечаток» ключа: ssh-keygen -l -f ~/.ssh/id_ed25519.pub
- Выбрать алгоритм хеша: добавить -E sha256 или -E md5

Цифровая гигиена: простые правила, которые спасут карьеру

Главная опасность — потеря или кража закрытого ключа. Снизить риски помогают элементарные привычки: сильная фраза-пароль, резервные копии в зашифрованном виде и разделение ключей по ролям. Не используйте один ключ для GitHub, продакшн-серверов и личного VPS — это как иметь один пароль для всех аккаунтов.

Для автоматизации храните открытые ключи в системе управления конфигурацией (Ansible, Salt), а закрытые — только на рабочих станциях разработчиков или в защищенном хранилище секретов типа HashiCorp Vault.

- Фраза-пароль минимум из 5-6 слов или 15+ символов
- Разные ключи для разных задач (работа, личные проекты, продакшен)
- Зашифрованные бэкапы ключей отдельно от основного устройства
- FIDO2-токены там, где это критично

Отзыв доступа: как правильно «выгнать» бывшего сотрудника

Когда кто-то увольняется или ключ компрометирован, доступ нужно отозвать быстро и без паники. Для обычных ключей просто удаляете соответствующую строку из authorized_keys на всех серверах. Хороший тон — вести список с указанием владельца каждого ключа и даты создания.

В больших организациях используют SSH-сертификаты: они живут ограниченное время и отзываются централизованно. Сложнее в настройке, но проще в управлении.

- Обычные ключи: удалить строку, завершить активные сессии пользователя
- Сертификаты: центральный отзыв через CA
- Детали формата authorized_keys: руководство sshd

SSH-сертификаты для продвинутых: когда ключей стало слишком много

Сертификат SSH — это обертка вокруг обычного ключа с подписью доверенного центра сертификации. Серверу достаточно доверять CA, а не каждому ключу по отдельности. Пользователю выдают короткоживущий сертификат с конкретными правами, что упрощает управление доступом.

Упрощенный пример:

```
# создаем ключ центра сертификации
ssh-keygen -f ~/.ssh/ssh_ca -C "Corporate CA"

# подписываем пользовательский ключ на 8 часов
ssh-keygen -s ~/.ssh/ssh_ca -I "developer-ivan" -n ivan,developer -V +8h
~/.ssh/id_ed25519.pub
```

Подробнее: раздел о сертификатах в ssh-keygen

Отладка: что делать, когда ничего не работает

90% проблем с SSH решается двумя способами: проверить права доступа к файлам и включить подробный вывод. Если SSH ругается на ключ, сначала посмотрите, что именно он пытается сделать:

```
ssh -vvv user@server.example.com
```

Частые проблемы и их решения:

- «Permissions 0644 are too open» — выполните chmod 600 ~/.ssh/id_*
- «No matching host key type» — сервер слишком старый, попробуйте RSA
- «Agent admitted failure to sign» — перезапустите агент и добавьте ключ заново
- Ключ правильный, но вход не работает — смотрите логи на сервере: /var/log/auth.log или journalctl -u sshd

Git и облачные сервисы: не только серверы любят SSH

Большинство Git-хостингов поддерживают аутентификацию по ключам. Добавляете открытый ключ в веб-интерфейсе, и можете клонировать репозитории, пушить коммиты и не вводить пароль каждый раз. Для изоляции лучше создать отдельный ключ «под Git».

Многие сервисы уже понимают ED25519, а некоторые поддерживают даже аппаратные ключи FIDO2:

- GitHub: подключение по SSH
- GitLab: документация по SSH
- Bitbucket: настройка SSH-ключей

Автоматизация: когда роботы создают ключи

В CI/CD-пайплайнах и скриптах важно детерминированное поведение без интерактивных вопросов. Все параметры можно задать флагами:

```
ssh-keygen -t ed25519 -a 100 -N "" -C "ci runner $(date +%Y-%m-%d)" -f
/tmp/id_ed25519_ci -q
```

Важно: пустая фраза-пароль (-N «») допустима только для временных ключей в изолированной среде!

- Закрытые ключи держите только в оперативной памяти
- Открытые ключи можно передавать через системы управления конфигурацией

Чек-листы для быстрой проверки

Настройка клиента:

- Создан ED25519-ключ с фразой-паролем
- Права: chmod 700 ~/.ssh && chmod 600 ~/.ssh/id_*
- Настроен ~/.ssh/config с IdentitiesOnly yes
- Агент запущен, ключ добавлен

Настройка сервера:

- ~/.ssh имеет права 700, authorized_keys — 600
- В sshd_config включен PubkeyAuthentication yes
- Сервис sshd перезапущен после изменений
- Сделан бэкап authorized_keys

Особенности платформ: Windows, macOS, Linux

На Linux и macOS OpenSSH входит в базовую поставку. В Windows 10/11 OpenSSH тоже есть из коробки, но служба агента может быть отключена. В WSL удобно держать ключи внутри подсистемы Linux, чтобы не путаться с правами доступа.

Современные IDE и Git-клиенты умеют работать с системным SSH-агентом, поэтому не нужно дублировать ключи в настройках приложений.

- Windows OpenSSH: официальная документация
- Проверьте настройки IDE — они часто используют системный SSH

Вопросы, которые все задают

- Можно ли использовать один ключ везде? Технически да, но лучше разделять по задачам. Потеря одного ключа не должна открывать доступ ко всему на свете.
- Обязательна ли фраза-пароль? Очень желательна. Без неё украденный файл ключа равен полной компрометации. С агентом вводить пароль нужно только один раз за сессию.
- Как понять, какой ключ сработал? Включите детальный вывод: ssh -v user@host. SSH покажет, какие ключи пробует и какой принял сервер.
- Что делать с потерянным ключом? Считать его скомпрометированным, удалить из всех authorized_keys, создать новый и добавить заново.

Полезные ссылки для закладок

- Мануал ssh-keygen — самый полный справочник
- Настройки ssh_config — все опции клиента
- Документация sshd — серверная часть
- GitHub SSH — подключение к репозиториям
- ssh-copy-id — копирование ключей на сервер

Выводы: SSH-ключи как спасение от парольного ада

Утилита ssh-keygen решает весь цикл работы с ключами от создания до отзыва. ED25519 с разумными настройками защиты подходит для большинства задач, аппаратные FIDO2-ключи — для особо критичных доступов. Остальное — вопрос организации: понятные комментарии, структурированное хранение файлов, единообразная конфигурация.

Один раз правильно настроив SSH-ключи, вы забудете о ежедневном вводе паролей к серверам и получите более высокую безопасность без постоянной головной боли. А если что-то пойдет не так, детальный вывод ssh -vvv и проверка прав доступа решают 99% проблем.

И помните: лучше потратить час на правильную настройку SSH, чем потом полдня разбираться, почему «оно не работает», особенно когда вся команда ждет деплоя в продакшен.

ssh, ssh-keygen, ssh-add, ssh-copy-id