# SSH

**SSH (Secure Shell)** - це криптографічний мережевий протокол, який дозволяє безпечно з'єднуватися з віддаленим сервером або комп'ютером через незахищену мережу, таку як Інтернет. Використовуючи SSH, можна виконувати різноманітні операції, такі як віддалене керування, передача файлів та шифрування з'єднання.

SSH забезпечує конфіденційність, цілісність та автентифікацію даних, що передаються між віддаленим сервером і локальним комп'ютером. Він захищає інформацію від перехоплення та зловживання з боку зловмисників, шифруючи дані під час передачі.

SSH також дозволяє виконувати команди на віддаленому сервері, надаючи доступ до командного рядка іншого комп'ютера з вашого локального комп'ютера. Це корисно для адміністрування серверів, виконання скриптів та взаємодії з віддаленими ресурсами.

Для підключення до віддаленого сервера через SSH, зазвичай використовується програма-клієнт SSH, яка доступні для різних операційних систем, таких як OpenSSH для Unix-подібних систем (Linux, macOS) і PuTTY для Windows. За допомогою цих програм ви можете ввести адресу сервера, ідентифікаційні дані (наприклад, ім'я користувача та пароль або ключ SSH) і встановити захищене з'єднання з віддаленим сервером.

# SSH – KEY GENERATION

SSH (Secure Shell) використовує ключі для шифрування та аутентифікації з'єднання. Для генерації ключів SSH можна використовувати різні варіації шифрування.

Основні варіації шифрування ключів SSH:

1. **RSA (Rivest-Shamir-Adleman)**: RSA є одним з найпоширеніших алгоритмів шифрування ключів SSH. Він використовує математичні принципи факторизації простих чисел для генерації ключів. RSA-ключі зазвичай мають довжину від 1024 до 4096 бітів.

2. **DSA (Digital Signature Algorithm)**: DSA є іншим алгоритмом шифрування ключів SSH. Він використовує математичні принципи дискретного логарифмування для генерації ключів. DSA-ключі зазвичай мають довжину 1024 бітів.

3. **ECDSA (Elliptic Curve Digital Signature Algorithm)**: ECDSA є алгоритмом шифрування ключів SSH, який базується на еліптичних кривих. Він забезпечує ту саму криптографічну силу, що й RSA та DSA, але з використанням ключів меншої довжини. Зазвичай використовуються ECDSA-ключі з довжиною 256 або 384 біти.

4. **Ed25519**: Ed25519 є новітнім алгоритмом шифрування ключів SSH, який базується на кривих EdDSA (Edwards-curve Digital Signature Algorithm). Цей алгоритм забезпечує високу криптографічну силу з ключами довжиною 256 бітів.

При генерації ключів SSH вибір конкретної варіації шифрування залежить від вашої конфігурації та вимог безпеки. RSA є найпоширенішим і розповсюдженим алгоритмом шифрування ключів SSH. Однак, варто

враховувати інші алгоритми, особливо ECDSA та Ed25519, які пропонують високу криптографічну силу з меншими ключами.

# ECDSA

**ECDSA (Elliptic Curve Digital Signature Algorithm)** є асиметричним алгоритмом шифрування та цифрового підпису, який використовує еліптичні криві для забезпечення криптографічної безпеки. Цей алгоритм базується на математичних принципах еліптичної криптографії.

Основні принципи ECDSA:

1. Еліптичні криві: ECDSA використовує математичні об'єкти, відомі як еліптичні криві, для генерації ключів і виконання операцій шифрування та цифрового підпису. Еліптичні криві мають особливі властивості, які забезпечують безпеку і ефективність алгоритму.

2. Ключі: У ECDSA використовуються пари ключів: приватний ключ і публічний ключ. Приватний ключ використовується для підпису повідомлень, тоді як публічний ключ використовується для перевірки підпису. Публічний ключ може бути розповсюдженим, тоді як приватний ключ повинен залишатися секретним.

3. Шифрування: ECDSA використовує математичні операції на еліптичних кривих для шифрування даних. Він забезпечує конфіденційність, тому що навіть якщо зловмисник отримає публічний ключ, він не зможе відновити приватний ключ і розшифрувати дані безпосередньо.

4. Цифровий підпис: ECDSA використовує приватний ключ для створення цифрового підпису повідомлення. Цей підпис може бути перевірений за допомогою відповідного публічного ключа. Цифровий підпис забезпечує цілісність та автентичність повідомлення.

ECDSA є безпечним і ефективним алгоритмом шифрування ключів, особливо з точки зору використання менших ключів порівняно з RSA. Він

широко використовується в різних протоколах, включаючи SSH для забезпечення безпечного з'єднання і цифрового підпису даних.

# Ed25519

**Ed25519** - це асиметричний алгоритм цифрового підпису, розроблений на основі кривих EdDSA (Edwards-curve Digital Signature Algorithm). Він отримав свою назву від параметрів, що використовуються в алгоритмі, зокрема, від кривої Ed25519, яка є частиною криптосистеми.

Основні характеристики Ed25519:

1. Криптографічна безпека: Ed25519 надає високий рівень безпеки при використанні ключів довжиною 256 бітів. Цей алгоритм забезпечує захист від атак на базі відомих проблем з шифруванням, таких як факторизація чисел або дискретний логарифм.

2. Ефективність: Ed25519 пропонує високу швидкість обчислень та підписування. Завдяки спеціальній конструкції кривої Ed25519 та оптимізованим арифметичним операціям, алгоритм є ефективним у порівнянні з іншими асиметричними алгоритмами.

3. Простота: Ed25519 має просту та ефективну конструкцію, що допомагає уникнути потенційних помилок у реалізації, що можуть призвести до безпекових проблем.

4. Аутентифікація: Ed25519 може використовуватися для створення цифрових підписів для підтвердження автентичності даних. Це дозволяє перевіряти, що дані не були змінені після підпису та що вони були створені особою з правом на це.

Ed25519 став досить популярним в останні роки і використовується в різних системах і протоколах для забезпечення безпеки та цифрового підпису, зокрема в системах електронної пошти, SSH, TLS, а також в блокчейн-

технологіях. Його швидкодія та безпека роблять його привабливим вибором для застосувань, які вимагають ефективного та надійного цифрового підпису.

# Параноїдальний ключ

**Параноїдальний ключ** - це термін, який походить від слова "параноя" і використовується для опису особливого виду криптографічного ключа, який генерується з високим рівнем випадковості та має дуже велику довжину.

Основна ідея застосування параноїдальних ключів полягає в тому, щоб зробити ключ настільки складним для вгадування або підбору, що навіть при використанні потужних обчислювальних ресурсів неможливо зламати або визначити його значення. Це робить параноїдальний ключ відповідним для захисту конфіденційності і цілісності даних.

Одним з відомих прикладів параноїдального ключа є ключ, згенерований за допомогою криптографічного алгоритму на основі хаотичних фізичних явищ, таких як шум електронних компонентів або рух частинок у квантових системах. Ці фізичні явища є непередбачуваними та випадковими, що забезпечує стійкість та неподатність параноїдального ключа до атак злому.

Параноїдальний ключ зазвичай має такі особливості:

1. Випадковість: Ключ генерується з великою кількістю випадкових бітів, які важко передбачити або повторити. Це робить його складним для підбору або зламу.

2. Довжина: Параноїдальний ключ зазвичай має достатньо велику довжину, щоб унеможливити його перебір. Часто використовуються ключі довжиною 128 біт або більше.

3. Криптографічна стійкість: Параноїдальний ключ повинен мати високу стійкість до криптографічних атак, включаючи атаки з використанням потужних обчислювальних ресурсів.

4. Використання криптографічних алгоритмів: Параноїдальний ключ зазвичай використовується для шифрування або підпису повідомлень за допомогою сильних криптографічних алгоритмів, таких як AES (Advanced Encryption Standard) або RSA (Rivest-Shamir-Adleman).

Загалом, параноїдальний ключ є дуже сильним і випадковим ключем, який забезпечує високий рівень безпеки у криптографічних операціях.

# Процес генерації ключів SSH

Процес генерації ключів SSH (**ssh-keygen**) використовується для створення пари ключів - приватного і публічного, які використовуються для безпечного з'єднання і аутентифікації на віддаленому сервері з використанням протоколу SSH.

Опис процесу генерації ключів SSH:

1. Виклик команди ssh-keygen: Ви викликаєте команду **ssh-keygen** на своєму комп'ютері.

2. Вибір розташування та назви файлів ключів: Буде запропоновано вибрати розташування та назву файлів для збереження ключів. За замовчуванням, приватний ключ зберігається в файлі `**id_rsa**`, а публічний ключ - в файлі `**id_rsa.pub**` у папці `**.ssh**` у домашній директорії. Можна залишити ці значення або вибрати власні.

3. Встановлення фрази-пароля (необов'язково): Буде запропоновано ввести фразу-пароль для захисту приватного ключа. Це додатковий шар безпеки, який шифрує ваш приватний ключ. Можна встановити фразу-

пароль або залишити її порожньою, але рекомендується встановити фразу-пароль для забезпечення безпеки ключів.

4. Генерація ключів: Команда **ssh-keygen** генерує випадкову пару ключів - приватний і публічний. Приватний ключ зберігається на вашому комп'ютері, а публічний ключ можна розповсюджувати на сервери або системи, до яких ви плануєте отримати доступ.

5. Завершення процесу: Після завершення генерації ключів, ви отримаєте підтвердження успішного створення ключів.

SSH-ключі забезпечують безпечний обмін даними та аутентифікацію на віддалених серверах з використанням криптографічних методів. Приватний ключ повинен залишатися в безпечному місці, тоді як публічний ключ можна розповсюджувати на сервери, до яких ви плануєте отримати доступ.

# Варіації шифрування ключів

У процесі генерації ключів SSH існує кілька варіацій шифрування ключів, які можна використовувати для забезпечення безпеки з'єднання SSH. Ось декілька варіацій шифрування ключів, які часто використовуються:

1. **RSA (Rivest-Shamir-Adleman)**: RSA є одним з найпоширеніших алгоритмів шифрування ключів. Він базується на складності факторизації великих простих чисел і використовується для генерації пари ключів - приватного і публічного.

2. **DSA (Digital Signature Algorithm)**: DSA є алгоритмом підпису, який також може використовуватися для генерації ключів SSH. Він забезпечує аутентифікацію та цифровий підпис повідомлень.

3. **ECDSA (Elliptic Curve Digital Signature Algorithm)**: ECDSA є алгоритмом підпису на основі еліптичних кривих. Він забезпечує ту

саму безпеку, що й RSA, але при меншому розмірі ключа, що робить його більш ефективним.

4. **Ed25519**: Ed25519 є алгоритмом підпису на основі едвардсової кривої. Він використовується для генерації ключів SSH і забезпечує високий рівень безпеки при невеликому розмірі ключа.

# Port forwarding

**Port forwarding**, також відомий як порт мапінг або тунелювання портів, є технікою мережевого налаштування, яка дозволяє пересилати мережевий трафік між двома різними мережевими інтерфейсами або комп'ютерами. Це корисний інструмент для забезпечення доступу до ресурсів або послуг, які знаходяться за мережевим брандмауером або маршрутизатором, а також для забезпечення безпеки мережі.

**Основна ідея порт форвардингу** полягає в тому, що ви пересилаєте мережевий трафік, який надходить на певний порт одного пристрою, на інший порт іншого пристрою. Це може бути використано в різних сценаріях, наприклад:

1. Remote Access: Ви можете налаштувати порт форвардинг для отримання віддаленого доступу до комп'ютера або сервера з будь-якого місця. Наприклад, ви можете перенаправити порт SSH на внутрішній сервер, щоб здійснити безпечне з'єднання.

2. Локальні послуги: Ви можете налаштувати порт форвардинг, щоб пересилати запити на певний порт до локальних послуг або програм, які використовуються на вашому комп'ютері. Наприклад, ви можете перенаправити запити на порт 80 на локальний веб-сервер для розробки або тестування веб-сайтів.

3. Безпека мережі: Порт форвардинг може бути використаний для забезпечення безпеки вашої мережі шляхом пересилання запитів на

спеціально налаштований проксі-сервер або мережевий пристрій, який може виконувати фільтрацію трафіку або обробку.

Порт форвардинг може бути налаштований на рівні мережевого пристрою (наприклад, маршрутизатора) або на рівні операційної системи (наприклад, за допомогою команди iptables у Linux або налаштування пробросу портів у Windows). Це важлива техніка для адміністрування мережі і забезпечення безпеки та доступності різних ресурсів.

# Типи Port Forwarding

Існує кілька типів перебросу портів, які можна використовувати для різних потреб:

1. **Local Port Forwarding** (локальний переброс портів): Цей тип перебросу портів дозволяє перенаправляти трафік з локального порту на віддалений порт через SSH-тунель. Запити, що надходять до локального порту, будуть пересилатися на віддалений сервер через SSH-з'єднання. Це корисно, коли ви хочете отримати доступ до служб або ресурсів, які розташовані на віддаленому сервері.

```
ssh -L 8080:192.168.0.100:80 user@192.168.0.100
```

2. **Remote Port Forwarding** (віддалений переброс портів): Цей тип перебросу портів дозволяє перенаправляти трафік з віддаленого порту на локальний порт через SSH-тунель. Запити, що надходять до віддаленого порту, будуть пересилатися на ваш локальний комп'ютер через SSH-з'єднання. Це корисно, коли ви хочете надати доступ до служб або ресурсів, які розташовані на вашому локальному комп'ютері через віддалений сервер.

```
ssh -R 8080:localhost:80 user@192.168.0.100
```

3. **Dynamic Port Forwarding** (динамічний переброс портів): Цей тип перебросу портів створює "динамічний" проксі-сервер, який дозволяє пересилати трафік з різних портів на віддалений сервер. Ви можете налаштувати ваш браузер або інший програмний засіб для використання цього проксі-сервера, що дозволяє вам анонімно та безпечно переглядати Інтернет або отримувати доступ до внутрішніх ресурсів через віддалений сервер.

```
ssh -D 8080 user@<адреса_віддаленого_сервера>
```

Ці типи перебросу портів дозволяють створювати з'єднання між різними комп'ютерами або серверами через SSH-тунель і перенаправляти мережевий трафік для різних цілей, таких як віддалений доступ, обмін даними або безпечне переглядання Інтернету.

# Практичне налаштування SSH

1. Встановлюємо пакет OpenSSH Server:

```
carrie@carrie-VirtualBox:~$ sudo apt update
```

```
carrie@carrie-VirtualBox:~$ sudo apt install openssh-server
```

2. Перевірка статусу:

```
carrie@carrie-VirtualBox:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: >
     Active: active (running) since Tue 2023-07-04 10:54:55 EEST; 38s ago
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 3181 (sshd)
      Tasks: 1 (limit: 2264)
     Memory: 1.7M
        CPU: 24ms
     CGroup: /system.slice/ssh.service
             └─3181 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

лип 04 10:54:55 carrie-VirtualBox systemd[1]: Starting OpenBSD Secure Shell se>
лип 04 10:54:55 carrie-VirtualBox sshd[3181]: Server listening on 0.0.0.0 port>
лип 04 10:54:55 carrie-VirtualBox sshd[3181]: Server listening on :: port 22.
лип 04 10:54:55 carrie-VirtualBox systemd[1]: Started OpenBSD Secure Shell ser>
lines 1-16/16 (END)
```

3. Налаштування файрволу(додаємо правило для порту SSH)

```
carrie@carrie-VirtualBox:~$ sudo ufw allow ssh
Rules updated
Rules updated (v6)
```

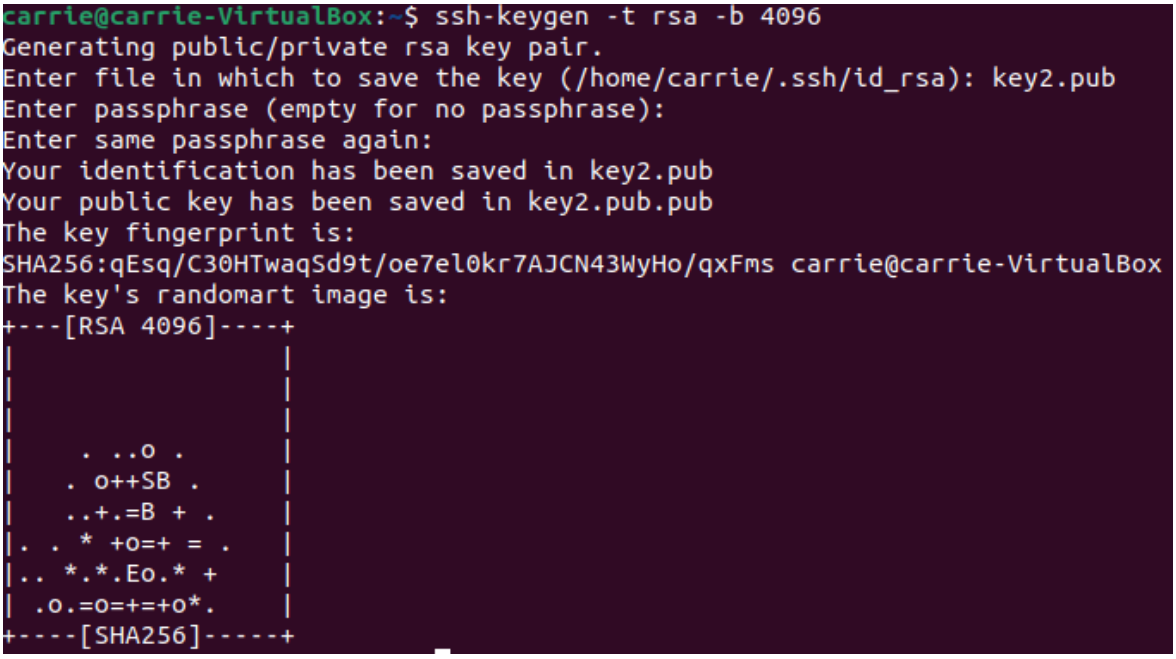
# Генерація ключів

1. Створення ключа за замовчуванням(RSA)

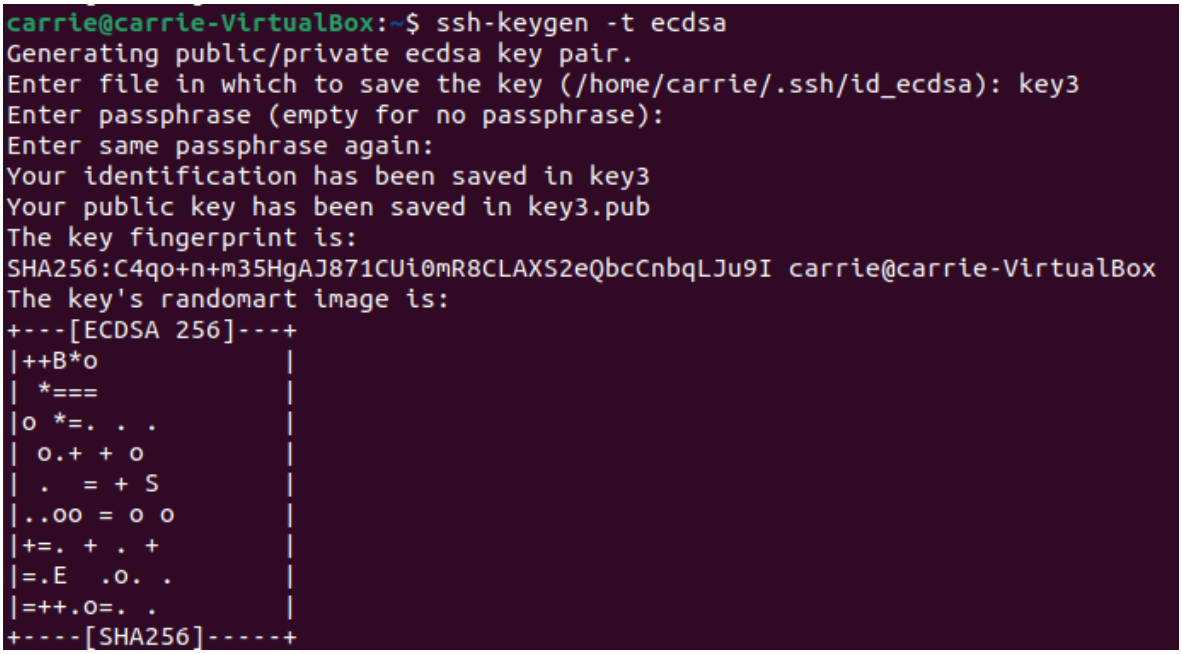
```
carrie@carrie-VirtualBox:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/carrie/.ssh/id_rsa): key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in key
Your public key has been saved in key.pub
The key fingerprint is:
SHA256:oDudmzhtYG/HhfaHlTFk4HwLIucRdkxTTVey1QEJ4Wk carrie@carrie-VirtualBox
The key's randomart image is:
+---[RSA 3072]----+
|        oo==++=.B|
|       . =o.+..=.|
|      o + +E. .  |
|     . = o.oo.   |
|    .   S.  .+   |
|    oo .o . o    |
|   .o+oo o o     |
|    .o=oo o .    |
|    .+o.   .     |
+----[SHA256]-----+
```

```
  GNU nano 6.2                                key.pub *
ssh-rsa AAAAB3NzaC1yc2
EAAAADAQABAAABgQC5E81oWJA/h99
XPqx5V4F8iF1k7oXMGYuzr5F5O+kJ
lPK5+49eqV5mKuzgrQdiAbClpTjM
obU8ptxnyTMUlDs8eF1pyhD5MnF
PDA/OTf8p7ZPXcOtrudbvHlRXob
XCsX0++ZHA5eIDVDxC6IV/3MPvl
C+BVvyFoMTx8SJOu2BWZQgOqY+\
w/8PZvrgtv26KBa1nH8dQOSxO8I
jj71cem42mnZ9qjE69mDEKYKyxb
Uk3GapKOZteLuFHbLrg1UCqNBGr
3/BvBbtWclzseurA9P1BcVrOt3/
OIlnZGXWJAkaTpz+tSMqHkNvew
MPHuvtv7Xw8q/PbZBBDeaLuYlP
NX3Mz7cd+vdcNsFcrrf2THvRT2
sm1uGziX7tsleVOvGZng6tKxf+Gn
yM7jhSmtZSn3Zko8DCMqElcuBp9z
gSfzZxrK3UlJFICZwK3llaVbt7zO
bBbcv9xImBIDi5svXUzX2GPuW6kl
4DZw3AqN40ANMUg/wcF9fh6y2m+/tjcEnR+DZM= carrie@carrie-VirtualBox
```

2. Створення ключа(RSA) зі специфічною довжиною:




3. Створення ключа за алгоритмом ECDSA



4. Створення ключа за алгоритмом Ed25519:

```
carrie@carrie-VirtualBox:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/carrie/.ssh/id_ed25519): key4
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in key4
Your public key has been saved in key4.pub
The key fingerprint is:
SHA256:yX5nVmWQRJayIuBIWbLaRA6f0Jw59l4Nwv/jDHNx5II carrie@carrie-VirtualBox
The key's randomart image is:
+--[ED25519 256]--+
|  ooo*.      o=o |
|   *X++ .   o.o. |
|   o*= + + o o  o|
|   +. o.E.= +  o |
|  . .. .So =  .  |
|      ..o +  .   |
|        .*..+    |
|         .o+     |
|                 |
+----[SHA256]-----+
```

5. Створення параноїдального ключа(RSA):

```
carrie@carrie-VirtualBox:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/carrie/.ssh/id_rsa): key5
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in key5
Your public key has been saved in key5.pub
The key fingerprint is:
SHA256:jU0G2eEDAl+czwB7j5zAiYuJ0erAOoHkv5ehwwKu5HE carrie@carrie-VirtualBox
The key's randomart image is:
+---[RSA 4096]----+
|    ..oo++..     |
| .   + ==+.      |
|... . * .+=      |
|=+ o . + Xo.     |
|*oo .   S +      |
|=..  .           |
|=+.E. o          |
|o+o+.o           |
|o...o            |
+----[SHA256]-----+
```


**Висновок:**

У даному контексті, генерація ключів SSH та налаштування SSH на операційних системах, таких як Ubuntu або Debian, є важливими аспектами для забезпечення безпеки з'єднання та аутентифікації на віддалених серверах.

Генерація ключів SSH включає створення пари ключів - приватного і публічного. Приватний ключ залишається на локальній машині, тоді як публічний ключ передається на віддалений сервер. Ці ключі використовуються для аутентифікації та забезпечення безпеки з'єднання. Налаштування SSH на операційній системі Ubuntu або Debian включає встановлення пакету OpenSSH Server, який надає можливість приймати з'єднання SSH. Також можуть бути необхідні додаткові кроки, такі як налаштування файрволу або зміна налаштувань конфігураційних файлів SSH.

SSH також підтримує різні типи перебросу портів, включаючи локальний, віддалений та динамічний переброс портів. Ці можливості дозволяють забезпечити доступ до різних сервісів та ресурсів на віддалених серверах через безпечне з'єднання SSH. В цілому, генерація ключів SSH та налаштування SSH є важливими процедурами для забезпечення безпеки та

захищеності з'єднання, а також забезпечення автентифікації на віддалених серверах. Ретельна увага до цих аспектів може значно покращити безпеку вашої системи та даних.