

## How it works

- Passwords are security measures to prevent unauthorized access to accounts, locks, and devices.
- A password may be stored on a computer or device in a way that is not readable by people. If hackers break into your computer, they won't find your cleartext password. Instead, they see an encrypted version of the password. When an account is created, the **cleartext** password is scrambled by a mathematical hashing function that cannot be reversed, such as **SHA-256**. The output of a hashing function is the **hash** and is 256 bits in length. The hash is stored on the micro:bit as a 64-character string in a file named "password.txt".
- During login, an **authentication** process is used to verify if a user's password is correct. During authentication, the hash saved in the "password.txt" file is read from the micro:bit and compared to the hash function output of the password just entered. If the two are equal, the user is granted access.
- A hacker can't use a password's hash like the cleartext password because, during authentication, the hash would be re-hashed and fail authentication.

## What will you do?

1. Practice hashing:
  - a. Advance to the page with "practice\_hash.py," run the program, and enter a password to display the hash string. Note – The string is 64 characters or 32 pairs; each pair represents a two-digit hexadecimal number; for example, "e4" represents the single byte 0xe4. 32 bytes x 8 bits/byte = 256 bits.
2. Set password to micro:bit:
  - a. Advance to the page with "set\_password.py" and run the program to hash your password and save the hash string to the "password.txt" file on your micro:bit. Note – the hash is a 64-character string; each character requires one byte of memory when stored on the micro:bit.
3. Read hash from micro:bit:
  - a. Advance to the page with "read\_password\_hash.py" and run the program. The file named "password.txt" will be read from the micro:bit and displayed. The hash string stored in 2a is read and displayed on the calculator. Note – the string is **not the cleartext password**.
4. Practice authentication:
  - a. Advance to the page with "authentication.py" and run the program to test your password and the authentication routine. This program compares the hash stored on the micro:bit to the hash of the entered password. If the two are equal, the user is granted access.
5. Remote login:
  - The **receiver**:
    - **Whisper the password of YOUR micro:bit to the sender.** Advance to 'student\_receiver.py,' change the group to your assigned number, and then run the program **before** the sender has run theirs.
  - The **sender**:
    - Advance to 'student\_sender.py,' change the group to your assigned number, and run your program **after** the receiver and hacker have started theirs. Enter the **password of the receiver's micro:bit**. You will be attempting to log in to the receiver's micro:bit.
  - The **hacker**:
    - Advance to 'student\_hacker.py,' change the group to your assigned number, and then run the program **before** the sender has run theirs.
  - Discuss among the group what message was sent over the radio. Can the hacker use the message they received to authenticate the receiver's micro:bit? Change roles and try again.

### Code it

#### Sender role

```
student_sender.py saved successfully
from microbit_radio import *
from hashing import *
# password must be kept private!
channel = 1
group = 1
clear_history()
password = input("Enter password: ")
password_hash = sha_hash(password)
tx(password_hash,channel,group)
```

#### Receiver role

```
student_receiver.py saved successfully
from microbit_radio import *
from hashing import *
# password must be kept private!
channel = 1
group = 1
clear_history()
test_hash = rx(channel,group)
authentic_hash = read_file("password.txt")
if test_hash == authentic_hash:
    display.show(Image.YES)
    music.play(music.BA_DING)
```

#### Hacker role

```
student_hacker.py saved successfully
from microbit_radio import *
from hashing import *
# password must be unknown to hacker

channel = 1
group = 1
clear_history()
hacked_hash = rx(channel,group)
print ("Hacked hash:",hacked_hash)
```

### Go further

- Run the “authentication.py” program again and enter the wrong password three times in a row.
- Try remote login again, but send the hash message the hacker received instead of the cleartext password.
- Try a different role in your team.

### Check your understanding

- The password hash, not the cleartext password, is saved on the micro:bit in the file named “password.txt.”
- A hacker can’t use the hash as the password during login; it would be re-hashed and fail authentication.
- A hashing function is a mathematical encryption. It is one-way; in other words, it cannot be reversed.
- A unique password has one, and only one, hash.

### Help

- Check that everyone on the team is using their assigned group number.
- Ensure the receiver and hacker run their programs and wait before the sender transmits the message.
- Ensure passwords are successfully set on each micro:bit.