



How it works

- A password is stored on a computer in a way that is not readable by humans. If a hacker scans your computer, they won't see your cleartext password. Instead, they see an encoded version called a hash.
- When an account is created, the cleartext password is encrypted by the hashing function, such as SHA-256, and stored in a file for later comparison during password authentication. The hash produced by SHA-256 is 256 bits in length.
- During password authentication, the entered cleartext password is hashed and compared to the valid hash recalled from the micro:bit. If the hashes are equal, the user is authenticated and granted control of the attached lock.
- During lock remote control, the hash is transmitted and may be vulnerable to eavesdropping. If a hacker gets the hash of your password, they can search for the hash in a rainbow table. If found, the cleartext password associated with that hash can be discovered.
- The micro:bit has several I/O pins that can control devices like a servo motor. The servo is a special motor that does not spin like a conventional motor but sweeps through an arc of 180° instead. In this activity, the servo moves the latch of the treasure chest to lock and unlock the lid.

What will you do?

1. Refer to the “Treasure Chest Build Instructions—Servo Motor.pdf” to assemble the circuit, 3D print the chest, and assemble.
2. Practice controlling the servo circuit
 - a. Advance to the page with “servo_test.py” and run the program to test the servo control circuit **before closing the lid**. Pressing the [enter] key will rotate the servo to the lock and unlock positions. An X or a ‘✓’ will be displayed on the micro:bit. Press the [esc] key to quit the program and, for safety, disconnect the battery until needed again. Note: if the servo buzzes when locked, refer to “Pick the Lock Parts and Assembly” to adjust the stop positions in the “servo_configuration.py” program.
3. Set a password on your micro:bit.
 - a. **Choose one of the ten common passwords** included in the rainbow table of this activity.

Abby	Adam	Beth	Burt	Cora
Abel	Brad	Bret	Carl	Dana

- b. Advance to the page with “set_password.py” and run the program to set the password of your choice on your micro:bit.

4. Practice unlocking and locking the treasure chest
 - 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; the servo is rotated 90° to open the chest latch.
 - b. Once the chest unlocks, press [enter] to rotate the servo back to the locked position.
5. Remote login to the treasure chest
 - The **receiver**:
 - Advance to the page with “student_receiver.py” and run the program. Share your password with the sender. They will open your treasure chest remotely.
 - The **sender**
 - Advance to the page with “student_sender.py.” Send the receiver’s password since you will remotely unlock their treasure chest. Run your program **after** the receiver and hacker have started theirs.
 - The **hacker**
 - a) Advance to the page with “student_hacker_brute.py” and run the program. You should receive the transmitted hash of the receiver’s treasure chest password. Use the brute_force module and run the function brute() to crack the password corresponding to the stolen hash. To use the function, type **brute[hacked_hash]** in the Python shell at the >>> prompt on the next page. Hint – use the Nspire’s [var] key to select the variable “hacked_hash” from a menu.
 - b) Advance to the page with “student_hacker_rbt.py” and run the program. You should receive the transmitted hash of the receiver’s treasure chest password. Use the rainbow table 'rbt' to look up the password corresponding to the stolen hash. To use the table, type **rbt[hacked_hash]** in the Python shell at the >>> prompt on the next page. Hint – use the Nspire’s [var] key to select the variable “hacked_hash” from a menu.
 - Once the hacker knows the receiver's cleartext password, the receiver should rerun their program. Next, the hacker should advance to the ‘student_sender.py’ program, run it, and enter the cleartext password of the hacked hash. Could the hacker open the lock without being told the receiver’s secret password?
 - Compare the hacker’s use of the rainbow table to the brute force attack. Which was the fastest? Which does not always return a result? Can you explain why?

Code it

Sender role

```

3.1 4.1 5.1 6 - Cyber...ock RAD X
student_sender.py 1/12
from microbit_radio import *
from hashing import *
# The sender must use the password of the
# micro:bit attached to receiver's calculator.
channel = 1
group = 1
clear_history()
password = input("Enter password: ")
password_hash = sha_hash(password)
tx(password_hash,channel,group)

```

Receiver role

```

2.2 3.1 4.1 6 - Cyber...ock RAD X
authentication.py 1/29
from microbit import *
from hashing import *
from servo_configure import *

clear_history()
input("Close chest and press [enter] to lock")
lock()
attempts = 0
for n in range(3):
    password = input("Enter password: ")
    test_hash = sha_hash(password)

```

Hacker role

```

5.1 5.2 5.3 *6 - Cybe...ock RAD X
student_hacker.py saved successfully
from microbit_radio import *
from rainbow_table import *
from servo_configure import *

channel = 1
group = 1
clear_history()
hacked_hash = rx(channel,group)
print("hash string = {}".format(hacked_hash))
# On the next page type rbt[hacked_hash]
# at the Python shell prompt >>>.

```

Go further

- Try a different role in your team.
- Try a new password that is not in the rainbow table.
- Change the sound and LED displays used during authentication.
- Change the servo to a different port; change the 'pin1' code to the corresponding pin number.

Check your understanding

- The micro:bit has input and output pins (I/O) that can be controlled in software. When a servo connected to an I/O pin is set to "100," the servo turns counterclockwise to the 100° position. When the pin is set to "0," the servo rotates clockwise back to the 0° position.
- The chest latch uses a servo with an attached horn to make the lock latch. When the servo is in the 5° position, the horn engages the latch, and when it is in the 100° position, the horn disengages the latch.
- The I/O pin that controls the servo is only accessible after passing authentication.
- A hash is a unique 256-bit string representing a cleartext password.
- Authentication compares a stored valid password hash with a calculated hash of an entered cleartext password. If the two hashes match, the system authenticates the user and grants access.
- During remote lock control, the password hash is sent to the receiver, not the cleartext password.
- A hacker can intercept a hash as it is transmitted from the sender to the receiver. The stolen hash can then be hacked using a rainbow table or a brute force attack. The hacker's rainbow table may work because it does not contain the intercepted hash. Alternatively, a brute force attack may require extremely long computing times which may prevent them from their dastardly deed.

Help

- Ensure the micro:bit Python module is installed on the calculator and the ti_runtime.hex is installed on the micro:bit card. These files are available at education.ti.com/microbit
- Try setting the password again and ensure you get 'file written and closed' on the calculator display and a '✓' on the micro:bit display.
- Use the "Treasure Chest Build Instructions—Servo Motor.pdf" to check the connections of your treasure chest.
- Ensure the external USB battery is charged and ON.
- Check that the servo is plugged into P1 on the Grove expansion shield.