



How it works

- A password is stored on a computer in a way that is not readable by humans. If hackers scan 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 or run a brute-force attack. If successful, the cleartext password associated with that hash is known.
- 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. Open ‘servo_7.py’ in the editor and run the program to test the servo control circuit. Press the [var] key and select “lock()” from the menu. This function will rotate the servo to 20°, and lock the latch when the horn is properly installed.
 - b. Press the [var] key and select “unlock()” from the menu. This function will rotate the servo to 150°, and unlock the latch when the horn is properly installed.
 - c. Repeat several times to test locking and unlocking the chest. If you hear a humming from the motor when locked, adjust the servo position as described in the “Treasure Chest Build Instructions—Servo Motor.pdf”
3. Set a password on your micro: bit using one of the ten commonly used passwords.

- Ten commonly used passwords:

Password	qwerty	111111	abc123	12345678
123456	guest	123123	123456789	12345

- a. Open 'set_pw_7.py' in the editor and run the program to set your selected password on your micro:bit.
4. Practice unlocking and locking the treasure chest
 - a. Open 'authen_7.py' in the editor 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 to open the chest latch.
 - b. Once the chest unlocks, close the lid and press [enter] to lock the chest.
5. Remote login to the treasure chest.
 - Ensure all group members use the same assigned group number.
 - The **receiver**:
 - a. Open 'recv_7.py', in the editor and run the program to program *before the sender*. Share your password with the sender. They will open your treasure chest remotely.
 - The **sender**
 - a. Open 'send_7.py' in the editor and run the program to 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. Open 'hack_7.py', in the editor and run the program *before the sender*. You should receive the transmitted hash of the receiver's treasure chest password.
 - b. Use the rainbow table 'rbt' to look up the password corresponding to the stolen hash. To use the table, type `rbt[stolen_hash]` in the Python shell at the `>>>` prompt on the next page. Hint – use the [var] key to select the variable "stolen_hash" from a menu.
 - Once the hacker cracks the receiver's cleartext password, the receiver should rerun their program. Next, the hacker should open the 'send_7.py' program in the editor, run it, and enter the cleartext password of the hacked hash of receiver's password.
 - 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

```

EDITOR: SEND_7
PROGRAM LINE 0001
from microbit import *
from mb_radio import *
from hashing import *
from mb_disp import *
from mb_music import *

radio.on()
radio.config(length=250, channel
            =12,power=6,group=1)
disp_clr()
pswd = input("Enter password: ")
  
```

Receiver role

```

EDITOR: RECV_7
PROGRAM LINE 0001
from microbit import *
from mb_radio import *
from mb_file import *
from mb_pins import *
from servo_7 import *

radio.on()
radio.config(length=250, channel
            =12,power=6,group=1)
disp_clr()
input("Close lid and [enter] to
  
```

Hacker role

```

EDITOR: HACK_7
PROGRAM LINE 0001
from microbit import *
from mb_radio import *
from mb_file import *
from rbt_5 import rbt
from brute_6 import brute

stolen_hash=""
radio.on()
radio.config(length=250, channel
            =12,power=6,group=1)
disp_clr()
  
```

Go further

- Try a different role in your team.
- Try setting a three-character password as in activity six and try a brute-force-attack.
- 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 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.

Files

- Transfer the activity files below to your calculator using the TI Connect CE Software. The link to download is [here](#). The best practice is to load all files for this cybersecurity activity and then delete them before loading the next set of activity files. This helps keep your calculator organized.

Name	Description
Servo_7.py	Tests and tunes the servo motor to properly lock and unlock the chest.
set_pw_7.py	It prompts for a three-lowercase character password and saves the hash in the ‘password.txt’ file on the micro:bit.
authen_7.py	Compares the hash of the entered password with the hash stored on the micro:bit.
send_7.py	Sends hashed password to receiver for authentication.
recv_7.py	Receives hashed password and authenticates.
hack_7.py	A man-in-the-middle attack snatches the hash from the sender and imports “rbt_5.py” to crack the stolen password hash.
rbt_5.py	Python dictionary with ten common passwords indexed by their hashes.
HASHING.8XV	SHA-256 hashing module

