CSAW ESC 2023 Final Report

Team: SH3L0CK, IIT-Madras

Arun Krishna AMS
Dept. of Electrical Engineering
IIT-Madras
Chennai, India
ee19b001@smail.iitm.ac.in

Antonson J
Dept. of Electrical Engineering
IIT-Madras
Chennai, India
ee19b025@smail.iitm.ac.in

Surya Prasad S
Dept. of Electrical Engineering
IIT-Madras
Chennai, India
ee19b121@smail.iitm.ac.in

Niranjan A Kartha
Dept. of Electrical Engineering
IIT-Madras
Chennai, India
ee21b095@smail.iitm.ac.in

Chester Rebeiro
Dept. of Computer Science & Engineering
IIT-Madras
Chennai, India
chester@cse.iitm.ac.in

***Abstract*—In the context of cyber-physical systems (CPS), the evolving landscape of security threats has prompted extensive research and countermeasure development. This paper investigates the side-channel attacks on the challenges provided in this competition involving running firmware on an Arduino microcontroller. A side-channel attack exploits additional information that can be acquired due to the implementation of a computer protocol or algorithm, rather than relying on inherent flaws in the protocol or algorithm's design.**

Each of these threats exploit distinct vulnerabilities within CPS, presenting unique challenges for mitigation. This paper explores the intricacies of the attacks and highlights countermeasures to defeat them.

***Index Terms*—Side-Channel Attacks, Timing Attacks, Constant-Time Programming, Audio-Side Channels, Confidentiality, Integrity, Authenticity**

I. Introduction

In an ever more interconnected world, cyber-physical systems (CPS) play a pivotal role in managing critical infrastructure, harmonizing the integration of digital control and physical processes. However, these systems are vulnerable to attacks that leverage unintentional information leakage channels, like timing, power consumption, and fault responses. These attacks can compromise the confidentiality, integrity, and availability of CPS.

As we rely increasingly on these systems, safeguarding them against emerging threats becomes a paramount concern for ensuring the smooth operation of essential services and infrastructure.

This paper provides the final report submission detailing the discoveries and accomplishments of our team – SH3RL0CK, IIT Madras, while analyzing the vulnerabilities present in the challenges of CSAW – ESC 2023.

II. *AllWhiteParty* Challenge

Username: Barry

Password: (Unknown)

A. Objective

The challenge requires both a username and password as input and computes hashed values to verify the password.

B. Deciphering Username

The username could be found using a timing attack. Timing attacks are a class of exploits that take advantage of the information inadvertently leaked by the system's response time to a given input.

The firmware seems to receive the input and compare characters one by one, starting from the beginning of the username, terminating when it encounters a differing character. As more characters match between the input and the username, the execution time increases.

Execution time was measured as the time it takes between the username input and the display of '*Invalid username. Please try again*'. For every character match, there was a 0.3-second increase in the time taken by the Arduino to output its result.

```
chars = ['a'..'z','A'..'Z','0'..'9'] # truncated
username = ""
prev_time = 0
for char in chars:
    #send username
    serial_write(username+char)
    start = measure_time()
    #receive Invalid username. Please try again
    serial_read()
    end = measure_time()
    time_taken = end - start
    if (time_taken > THRESHOLD + prev_time)
        username = username + char
        prev_time = time_taken
```

Listing 1. Timing side-channel exploit

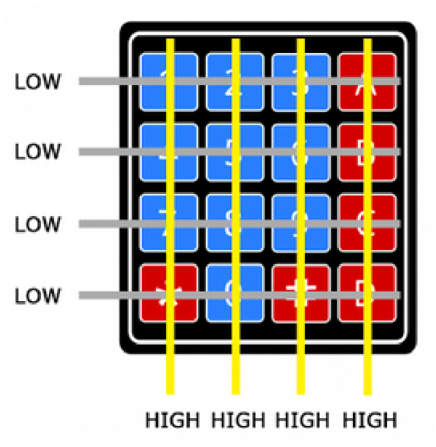


Fig. 1. When no button is pressed

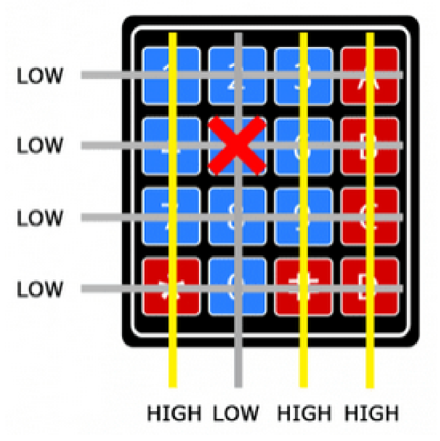


Fig. 2. When a key on column 2 is pressed

C. Mitigation

To mitigate timing attacks, one approach is to introduce random delays into functions. A more robust strategy involves ensuring that all possible branches of a function consume a consistent same amount of time to compute. This practice is known as constant-time programming and should be employed whenever a function handles sensitive information.

D. Deciphering Password

Unlike the username, which has to be entered through serial, the password has to be entered through a keypad manually. Our first task was to automate the entire username + password entry process.

The keypad communicates (digital signal) with the *PCF8574* I/O Expander board which communicates with the Arduino using *I2C*[3]. The Arduino detects which button is pressed by detecting the row and column pin that's connected to the button.

- When no buttons are pressed, all of the column pins are held HIGH, and all of the row pins are held LOW (Figure 1). When a button is pressed, the column pin is pulled LOW, and this is measured (Figure 2)
- Now all the row pins are held HIGH, while all of the column pins are held LOW. The row pin corresponding to the pressed button is pulled LOW, and this is measured.[2]

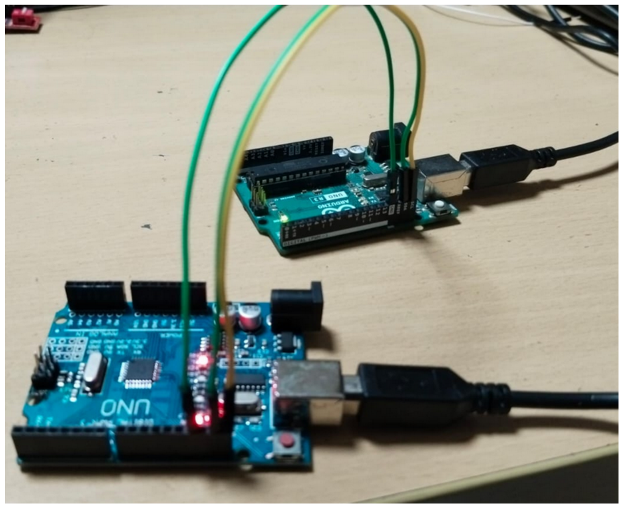


Fig. 3. AllWhiteParty Setup

To enable rapid testing, the entire process is automated, with the 'spy' Arduino acting as *I2C* slave instead of a *PCF8574*, to mimic the keypad. The slave address at which *PCF8574* communicates is observed at *0x20* (7-bit *I2C* address). We initially 'spied' the communication between *PCF8574* and Arduino using an *Analog Discovery 2 Digilent* oscilloscope to get familiar with the communication, but the process was automated later using an Arduino (Figure 3).

E. Observations

1) Observation 1: The following behaviour was identified and observed:

1) Firmware accepts usernames of the format "*Barry[xxxx...x]*", i.e., only the first characters are checked for equality with "*Barry*".
2) The displayed hash value depends only on the username and not on the password entered. For example:
   - *Username:* Barry
     SHA hash: 167 98 212 144 128
   - *Username:* Barry123
     SHA hash: 164 167 250 75 88
   - *Username:* Barry12345
     SHA hash: 113 71 16 189 178
3) Time taken to compute the hash was observed to be based on the number of characters added to the username. We calculated the time taken for the username "*Barry* + $\underbrace{\backslash 0\backslash 0\backslash 0\ldots}_{n\text{ times}}$" varying n from 0 to 650

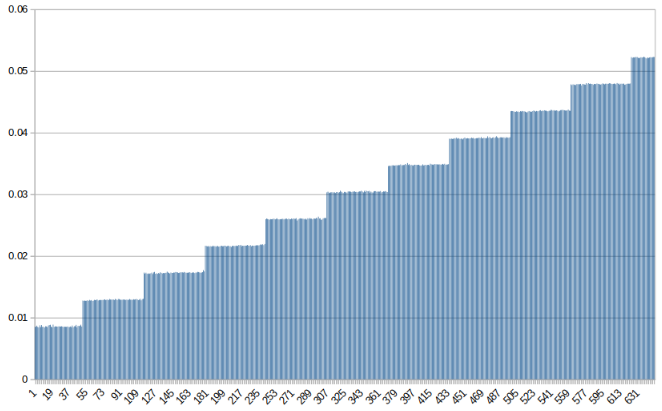


Fig. 4. Time taken vs *n*

- We observed that for $0 \leq n < 50$, time taken was approximately 8 milliseconds
- From $n = 50$ onwards, for every 64 bytes of characters added we observed that the time taken increased approximately 4 milliseconds i.e., (Figure 4)
  - When $0 \leq n < 50$: 8 milliseconds
  - When $50 \leq n < 114$: 12 milliseconds
  - When $114 \leq n < 178$: 16 milliseconds

Based on the above observation we conclude that:

1) The hashing algorithm used for hashing utilizes a 64-byte block-size
2) The hash algorithm takes around 4 milliseconds for every 64-byte block
3) Initially until $n < 50$, the algorithm takes 8 milliseconds. This can be attributed to the algorithm utilizing 5 [”*Barry*”] + 49 [n] + 10 [”*Password*”] = 64 bytes for input.

2) Observation 2: We observed that for the following pairs, the time taken for the Arduino to respond to our password was 0.3 seconds (much higher than usual 8 milliseconds).

- **Username: Barry, Password: 3805032704**
- **Username: Barry, Password: 3835032704**
- **Username: Barry, Password: 3801032702**
- **Username: Barry, Password: 3865062704**

This indicates an existence of timing-related side channels in the password comparison as well. Since the password is hashed first, before it is compared, we couldn’t exploit them.

III. *BlueBox* Challenge

Flag: B339B009

A. Initial Observations

1) A quick succession of notes is played on reset. This is different across different resets.

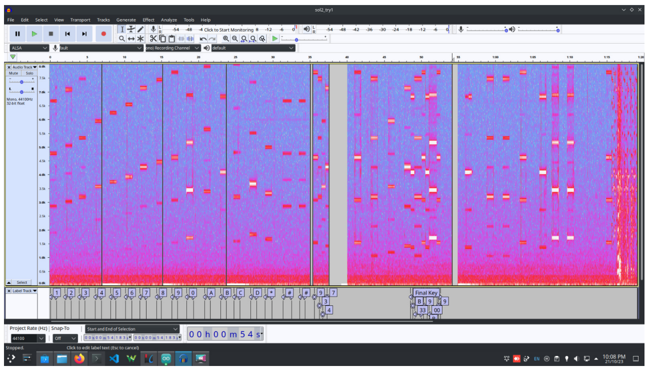


Fig. 5. Analyzing tones using Audacity

2) On pressing a key on the keypad, a unique note plays. This is always the same as long as the key pressed is the same i.e., the note ↔ key mapping is unique and constant.

This indicates that there exists an audio-based side-channel vulnerability in the system which is used to exploit to retrieve the flag.

B. Approach

The notes played corresponding to each key on the keypad were recorded, and frequency-analyzed using Audacity, an open-source audio software [1].

The note progression played on reset was recorded and analyzed through audacity to obtain the notes and match the known notes to the corresponding keys on the keypad. The keys were then entered into the keypad. This resulted in yet another series of notes playing. This was recorded and analyzed similarly, in order to get the final passcode.

C. Mitigation

- **Acoustic analysis** must be performed to understand how the device might leak information through audio-side-channels. A system must be designed such that it does not leak information during key-presses i.e., constant-same tone when any key is pressed.
- **Encryption**: Encrypt the audio output. If an attacker captures the audio signal, it will be unintelligible without the decryption key.
- **Frequency Hopping:** Implement a mechanism where the frequency used for each key press is randomly selected from a predefined set. This makes it more challenging to determine which key was pressed based on the frequency.
- **Noise Injection**:
  - Add random noise to the audio signal generated by the keypad
  - Slightly vary the frequency of the tones generated for each key
  - Randomly adjust the volume of the tones
  - Introduce random delays between the time a key is pressed and the corresponding tone is generated.

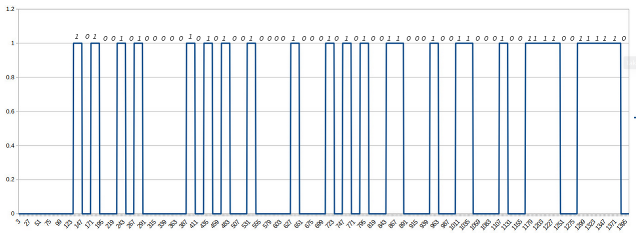


Fig. 6. Initial message: A5 05 48 45 4C 4C 4F 3E

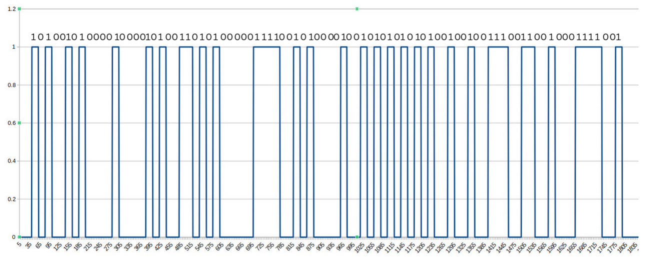


Fig. 7. Flag: A5 08 53 50 79 42 55 52 4E 64 79

- Apply low-pass filters to the audio output to reduce high-frequency components that might carry information

The last two measures - Frequency hopping and noise injection do not remove the threat vulnerability, but reduces the correlation between the audio-signals and key press events, thereby making it significantly more challenging to determine which key is pressed.

IV. *SPitFire* Challenge

Flag: SPyBURNd

A. *Initial Observation*

Relay connected to the Arduino keeps switching.

B. *Approach*

The state of the relay connected to the victim Arduino was monitored by utilizing digital pins on a secondary Arduino. The secondary Arduino was responsible for broadcasting the state of the digital pin over a Serial connection. This data was recorded in a file and later processed by a defined Python script.

As a result, an ASCII-encoded message was generated, with a header byte, length of the message (excluding length of header and checksum byte) and a final checksum byte employing *CRC8*. CRC was evaluated and confirmed using a Python Library.

C. *Messages*

- *Initial message observed at relay (Figure 6):*
```
A5 05 48 45 4C 4C 4F 3E
      H  E  L  L  O
```
- *Message sent serially*
```
A5 04 46 4C 41 47 DA
      F  L  A  G
```
- *Flag observed at relay (Figure 7):*
```
A5 08 53 50 79 42 55 52 4E 64 79
      S  P  y  B  U  R  N  d
```

D. *Mitigation*

Communication in the firmware happens in an open channel which does not ensure confidentiality, authenticity and integrity of the messages.

- **Confidentiality** ensures that information is only accessible to authorized individuals or systems. It involves protecting data from unauthorized access or disclosure. Common encryption algorithms like Advanced Encryption Standard (`AES`) & `RSA` can be used for providing encryption.

- **Authenticity** verifies the identity of users, devices, or entities involved in a communication or transaction. It ensures that the parties involved are who they claim to be. Digital signatures provide a way to prove the identity of the sender and verify that the data has not been tampered with. Asymmetric cryptography is often used for digital signatures (e.g., `RSA` or `ECDSA`).

- **Integrity** ensures that data remains unaltered and trustworthy throughout its life cycle. It involves protecting data from unauthorized modification, deletion, or corruption. To guarantee integrity, use a message authentication code `MAC` or a cryptographic hash function. A `MAC` or a hash function generates a fixed-size digest of the data, which can be used to detect any unauthorized modifications

V. *czNxdTNuYzM* Challenge

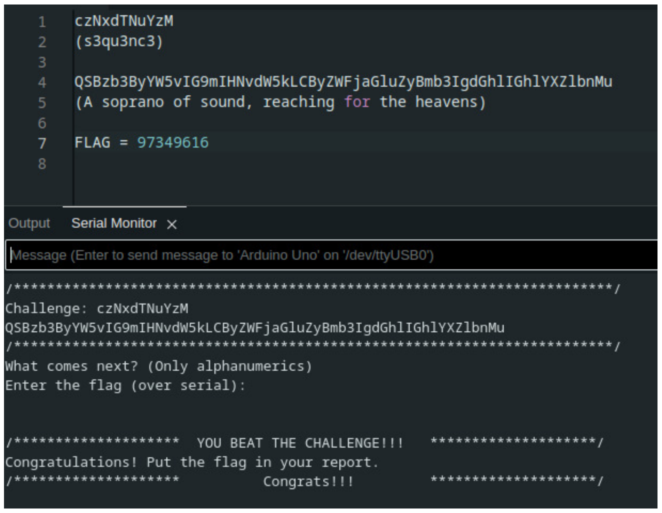
Flag: 97349616

A. *Initial observations*

On the 7-segment display, digits were being flashed very quickly. Text encoded in base64 is printed through Serial.

B. *Approach*

The 7-segment display was recorded in slow motion, and the digits were noted. In order to find the next number in the number sequence so obtained, the values were queried in the On-Line Encyclopedia of Integer Sequences (OEIS) [5]. This gave one matching sequence (*A008336*) which satisfied the initial values.



```
czNxdTNuYzM
(s3qu3nc3)

QSBzb3ByYW5vIG9mIHNvdW5kLCByZWFjaGluZyBmb3IgdGhlIGhlYXZlbnMu
(A soprano of sound, reaching for the heavens)

FLAG = 97349616
```

```
/**********************************************************************/
Challenge: czNxdTNuYzM
QSBzb3ByYW5vIG9mIHNvdW5kLCByZWFjaGluZyBmb3IgdGhlIGhlYXZlbnMu
/**********************************************************************/
What comes next? (Only alphanumerics)
Enter the flag (over serial):


/*******************  YOU BEAT THE CHALLENGE!!!   *******************/
Congratulations! Put the flag in your report.
/*******************           Congrats!!!        *******************/
```

Fig. 8. *czNxdTNuYzM* Output

Number Sequence: *1, 2, 6, 24, 120, 20, 140, 1120, 10080, 1008, 11088, 924, 12012, 858, 12870, 205920, 3500640, 194480, 3695120, 184756, 3879876, 176358, 4056234, 97349616, 2433740400*

C. *Mitigations*

Various techniques invented recently involve the use of video in order to obtain side-channel information. [4] Thus, light sources in the system which may be affected by various internal factors may become a side-channel for information, and care needs to be taken to mitigate them.

VI. *SockAndRoll* CHALLENGE

Flag: (Unknown)

A. *Initial Observation:*

The program claims to listen for a message during which a sequence of beeps is heard through the buzzer, after which the Arduino assumes that everything is OK.

B. *Approach*

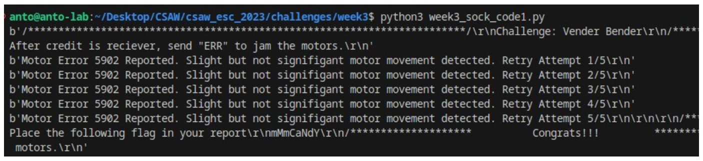
We initially hypothesized that the Arduino listens for the beeps from the buzzer as when the buzzer is removed a message gets displayed saying ”Unable to decipher the message”.

We observed that there are two different tones being played i.e., one audible – within human hearing range, whereas the other is inaudible.

C. *Observations*

This approach failed for the following reasons:

1) Even if the microphone was listening for beeps, the Arduino appears to only respond to beeps generated by itself. Recording the beeps on a phone and playing it back results in a “failed to interpret” message.
2) The assumption that the microphone listens for beeps is questionable. Sometimes, when we place the buzzer very far away such that the microphone could not physically perceive it, or ground one of its data pins, the Arduino still perceives that we are OK. In fact, as long as the buzzer and microphone are both connected, the Arduino seems to think that we are OK. We were not able to replicate this problem reliably.



```
anto@anto-lab:~/Desktop/CSAW/csaw_esc_2023/challenges/week3$ python3 week3_sock_code1.py
b'/**********************************************************************/\r\nChallenge: Vender Bender\r\n/*****
After credit is reciever, send "ERR" to jam the motors.\r\n'
b'Motor Error 5902 Reported. Slight but not signifigant motor movement detected. Retry Attempt 1/5\r\n'
b'Motor Error 5902 Reported. Slight but not signifigant motor movement detected. Retry Attempt 2/5\r\n'
b'Motor Error 5902 Reported. Slight but not signifigant motor movement detected. Retry Attempt 3/5\r\n'
b'Motor Error 5902 Reported. Slight but not signifigant motor movement detected. Retry Attempt 4/5\r\n'
b'Motor Error 5902 Reported. Slight but not signifigant motor movement detected. Retry Attempt 5/5\r\n\r\n\r\n/***
Place the following flag in your report\r\nmMmCaNdY\r\n/*******************         Congrats!!!        *******
 motors.\r\n'
```

Fig. 9. *VenderBender* Output

VII. *VenderBender* CHALLENGE

Flag: mMmCaNdY

A. *Initial Observations*

The relay connected to victim Arduino switches on-off at intervals, and the victim Arduino lets us send a glitch signal (send ‘ERR’ through Serial). The glitch signal must be sent at the right instant in order to obtain the flag.

B. *Approach*

A second Arduino was used to observe the relay state and transmit the state over Serial. This output was observed by a Python script, which would send the glitch signal (send ‘ERR’ through Serial) as soon as it came across a rising edge in the relay state. On repeatedly thrashing the system at approximately the right instance, the fault is injected and the flag is retrieved.

```
attachInterrupt(pin, RISING_EDGE, callback_function)
function callback_function():
    serial.write('E')
```

Listing 2. Arduino code

```
while True:
    if serial_spy.read() == 'E':
        serial_victim.write('ERR')
```

Listing 3. Python code in PC

C. *Mitigation*

The following measures can be implemented to help protect a system from fault injection attacks:

- All the backdoor entries to the system must be closed before the firmware is packaged.
- **Fuzzing:** (Fuzz testing) must be done to discover any potential faults and vulnerabilities in the system. Fuzzing tools generate large volumes of input data, often with

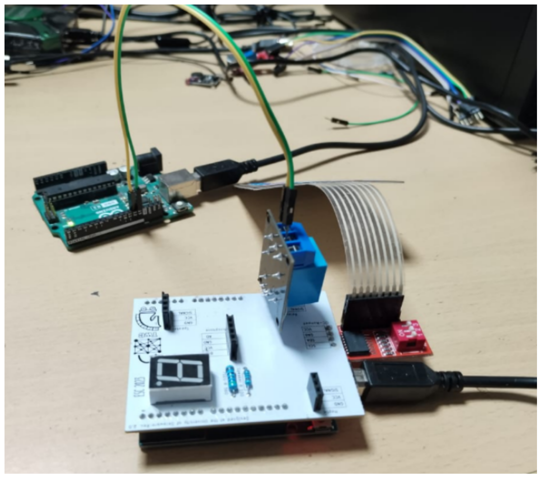


Fig. 10. *VenderBender* setup

the intent of causing buffer overflows, crashes, or other unexpected behaviors in the target software. Fuzzing results in crash reports and input cases that triggered unexpected behavior in the software.

VIII. Conclusion

In this report, we demonstrated that when the hardware and software is not protected against side channel attacks, even simple techniques may be sufficient to obtain unauthorized access to embedded systems.

As novel attack vectors emerge, ongoing research and development efforts are essential to remain ahead of potential threats. This includes not only improving existing defenses but also exploring techniques that reduce the tradeoff between security and system performance.

Documents

Following documents can be in the drive link:

- *AllWhiteParty* Challenge:
  - Python code to obtain username: `sol1a_get_username.py`
  - Python and arduino code to automate username - password entry for keypad: `automate_password.py` & `arduino_replicate_pcf8574.ino`
  - Data used to plot graph in *Figure 4*: `store_time_q1a1.ods`
- *BlueBox* Challenge:
  - Audacity Project files: `audacity_bluebox.aup` & `audacity_bluebox.zip`
- *SPitFire* Challenge:
  - Relay input when *"HELLO"* was sent: `hello_relay_data.txt`
  - Relay input when *flag* was sent: `flag_relay_data.txt`
  - Python code to parse through the data: `read_data_w2q1.py`
- *czNxdTNuYzM* Challenge:
  - Video files: `lcd_display1.mp4` & `lcd_display2.mp4`
- *VenderBender* Challenge:
  - Arduino code: `vender_detect_relay.ino`
  - Python code: `vender_send_glitches.py`

Acknowledgment

We would like to express our sincere gratitude to Prof. Chester Rebeiro for his invaluable guidance and support throughout this research phase. Furthermore, we acknowledge the support and resources provided by IIT-Madras, which facilitated our research endeavors and participation in CSAW 2023. We are grateful to our colleagues and friends for their feedback and encouragement.

Finally, we extend our thanks to the organizers of the CSAW 2023 for the opportunity to present this work under the research track.

References

[1] Audacity, free open-source audio software. https://www.audacityteam.org/.

[2] How to set up a keypad on an arduino. https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/.

[3] Pcf8574 remote 8-bit i/o expander for i2c bus datasheet. https://www.ti.com/lit/ds/symlink/pcf8574.pdf.

[4] Ben Nassi, Etay Iluz, Or Cohen, Ofek Vayner, Dudi Nassi, Boris Zadov, and Yuval Elovici. Video-based cryptanalysis: Extracting cryptographic keys from video footage of a device's power led. Cryptology ePrint Archive, Paper 2023/923, 2023. https://eprint.iacr.org/2023/923.

[5] Neil J. A. Sloane and The OEIS Foundation Inc. The on-line encyclopedia of integer sequences, 2023.