CSAW Final Paper
Yellow Hackets

Shayan Aqeel
Georgia Institute of Technology
Atlanta, Georgia
saqeel3@gatech.edu

Henry Bui
Georgia Institute of Technology
Atlanta, Georgia
hbui43@gatech.edu

Tracy Guo
Georgia Institute of Technology
Atlanta, Georgia
tguo72@gatech.edu

Smit Patel
University of North Georgia
Dahlonega, Georgia
snpate8729@ung.edu

I. Introduction

This paper describes all of the work that the Yellow Hackets team completed within the semester competing in the Cybersecurity Awareness Embedded Security Challenge (CSAW ESC).

The CSAW ESC is a challenge where teams are given an Arduino acting as a cyber-physical system and tasked to crack and expose keys within the system. For the 2024 season, the CSAW ESC specifically focused on manufacturing-based cyber-physical systems and possible exploits that could be performed in a manufacturing environment.

This paper focuses on the research done to prepare for the challenge (particularly the possible exploits and side-channel attack vulnerabilities that could be performed against a manufacturing-based cyberphysical system), the technical work done to tackle each of the challenges in the CSAW ESC, and individual contributions of each of the team members.

II. Research on Side-Channel Attacks

Before competing in the ESC, the team needed to know and understand what side-channel attacks were and which types would be applicable to the type of embedded systems that the team would be attacking during the competition. As such, the team looked over potential approaches and possible attack methodologies for side-channel attacks against manufacturing-based cyber-physical systems. The team also looked into possible mitigation strategies that could be applied to manufacturing systems.

A. Potential Approaches

Given this year's ESC focus on SCAs targeting Arduino-based cyber-physical systems (CPS) in manufacturing processes, the following methods outline how these vulnerabilities can be exploited and their potential impact on manufacturing systems.

1) Timing Analysis: One approach that is used to attack cyber-physical systems (and other systems susceptible to side-channel attacks) is a timing attack [1]. Certain operations take more time to complete during a computation than others (for example, the multiplication of large numbers will take much longer than the addition of large numbers). Thus, a timing attack takes advantage of this by analyzing the differences in time required to complete an operation in a given algorithm implementation of a cryptosystem. While simple, this can still be quite effective. If different key patterns result in distinct and discernible patterns in timing, then this can be exploited to partially or fully reverse-engineer a private key.

2) Energy Consumption Analysis: Another approach is to analyze the power consumption of the CPS, allowing researchers to collect details about the manufacturing process. Measuring power at various manufacturing stages enables attackers to determine what is being manufactured or the complexity of the task. Furthermore, this technique leverages the differential power analysis (DPA) method, where variations in power consumption are statistically analyzed to uncover patterns linked to the system's operation [2]. For example, Kocher et al. [2] have shown that such a technique can help reverse engineer sensitive details like cryptographic keys.

3) Disruption Through Timing Manipulation: An attacker may also manipulate the timing of operations. By introducing intentional delays or tweaking the timing of critical tasks, attackers disrupt the synchronization of the system and cause errors or inefficiencies in the process. This is known as a timing attack, as the attack introduces deliberate interference in the timing variations that occur during operation, which can snowball into significant disruptions [1].

4) Acoustic Side-Channel Attacks: Another approach is an acoustic side-channel attack, where the audio/sound emitted by a cyber-physical system during the manufacturing process is analyzed to understand and reverse-engineer operational information.

Al Faruque et al. [3] designed a model to exploit this idea. They reconstructed an object created from an additive man-

ufacturing cyber-physical system (in particular, a 3D printer) by analyzing the sound of the nozzle moving in the printing process. These models consisted of a regression model to predict the speed of the nozzle from the frequency of the sound of the nozzle's motion and a classification model to predict the axis (x, y, z) of the nozzle from the sound of the motor. This allows an attacker to reverse-engineer the G-code (specification) for how an object is designed and utilize that to create copies of their own.

While making copies of objects may not seem like the most severe security issue, it is quite a significant security issue due to copying being an act of stealing a given manufacturer's intellectual property. Additionally, with parties like the Air Force, Navy, and NASA taking advantage of additive manufacturing cyber-physical systems, it is vital to protect this intellectual property.

5) Electromagnetic Emission Analysis: One last approach for attacking cyberphysical systems is utilizing the electromagnetic emissions from manufacturing components as a side channel. In this case, researchers can use tools such as spectrum analyzers to capture electromagnetic signals correlating to the system's operations [4]. This then reveals sensitive information about the CPS manufacturing operation.

B. Attack Methodologies

1) System Analysis: To properly exploit vulnerabilities in a cyber-physical system, it is necessary to understand the system's operations and its interactions with the outside world. First, by recognizing the critical functions of a system and its intended behaviors, one has a foundation to disrupt the device. Then, one can realize potential vulnerabilities in the physical components by identifying physical components, such as controllers, chips, sensors, and actuators. Finally, one can obtain access to the system's operations by gathering critical system information—such as cryptographic keys and control commands—one. Thus, a thorough analysis of the system clearly leads to data collection and potential attack vectors.

2) Data Collection: There are many different tools that can be used to collect side-channel data on the system. These include oscilloscopes (for measuring power consumption), spectrum analyzers (for measuring electromagnetic emissions), high-precision timers (for timing analysis), microphones (for measuring sounds and vibrations), and thermal cameras (for heat patterns) [5]. These measurements can be performed while the system operates normally, or they can be performed whilst deploying intentional faults in system operation (known as a fault injection). Intentional faults by varying factors such as power supply, clock frequency, electromagnetic emissions and temperature can give additional information to use based on how the system may respond [6]. Pairing all this data with known system elements can help determine potential side-channel attack technique candidates.

3) Data Analysis and Testing: Once side-channel data about the system is gathered, one can analyze patterns and correlations in the data to determine the best SCA technique to attempt. Differential fault analysis (DFA) can be utilized where differences in the differential path versus the expected path when analyzing results of faulty and faulty-free injections can eliminate potential keys. Fault sensitivity analysis (FSA) can also be performed to measure how calculations done by the system respond to varying degrees of fault intensity [6]. Furthermore, machine learning and deep learning models can be employed with well-crafted datasets to find these correlations and patterns more readily. Finally, once valid patterns are found and a side channel attack technique is successfully deployed, the cyber-physical system can be used for the attacker's purposes.

C. Mitigations of Manufacturing SCAs

Mitigating the impact of SCA for manufacturing CPS include techniques to reduce sensitive information from leaking. The goal is to ensure attackers cannot exploit physical emissions like power consumption, timing, or sound to reverse-engineer or disrupt manufacturing processes.

1) Random Delays and Random Noise: One mitigation is implementing delays to deter timing attacks. Delays can be implemented with hardware circuitry [7]. This forces the attacker to collect more measurements, making detecting patterns that reveal sensitive information harder. However, since a differential power analysis attack can exploit noise alone, noise injection should be combined with other countermeasures to offer multilayered protection.

2) Power Line Conditioning and Filtering: Filtering the power supply can prevent attackers from correlating power consumption with sensitive operations, such as encryption. This prevents the attacker from analyzing power traces to extract sensitive information [7]. This countermeasure is also implemented alongside others to provide comprehensive system protection.

3) Binding and Masking: Binding and masking alter the input of cryptographic algorithms or split sensitive variables into multiple shares to obfuscate their values [4]. These cryptographic countermeasures reduce the effectiveness of SCAs by ensuring that the leaked information is unrelated to the actual sensitive data. Blinding makes it difficult for attackers to analyze the leaked data and masking prevents sensitive values from being put back together.

4) Hardware-Based Countermeasures: Hardware-based countermeasures can be integrated into the manufacturing CPS. They can secure the hardware itself and prevent SCAs by embedding protections that obfuscate the correlation between physical emissions (like power consumption and electromagnetic signals) and sensitive data.

One hardware-based countermeasure is *power equalization*. In power equalization, the hardware can be designed to consume a constant amount of power instead of allowing various power consumption based on the operations being performed [4]. This removes the correlation between the power released and the data being processed. If the hardware is processing sensitive data or performing routine operations, the power profile remains constant, which prevents attackers from using differential power analysis to extract information.

Another hardware-based countermeasure is *power shielding and electromagnetic noise suppression*. Here, the manufacturing of CPS can include physical shielding that reduces the amount of EM radiation released from the device [4]. It could also emit extra electromagnetic noise to mask signals. These measures make it challenging for an attacker to isolate and analyze EM emissions related to operations within the manufacturing system.

III. Open-Source Intelligence on Arduino Uno R3

Beyond doing general research on SCAs targeting cyber-physical systems in manufacturing processes, the team also performed open-source intelligence on the Arduino Uno system that this year's ESC is built upon.

The team found that the Arduino Uno R3 is a microcontroller board which consists of an ATmega328P chip, 14 digital input/output pins, (six of which can be used as Pulse Width Modulation (PWM) outputs), six analog inputs, a 16 MHz ceramic resonator, a USB connection for programming, a power jack, an ICSP header for in-system programming, and an EEPROM memory module. The team discussed possible ways of exploiting these items, including reading unprotected data from an EEPROM with a malicious program and analyzing timing signals from the ATmega328P chip in order to extract private keys.

Furthermore, the team analyzed research papers and textbooks to understand the usage of Arduino within embedded systems, specifically for the use of cyber physical systems (CPS). Which helped them understand the building blocks of low cost micro controller within a CPS environment. Using this information the team was able to understand that during the competitions challenges they could debug the code given to help them solve the challagnges.

Also, during this time the team looked into open-source models build for CPS and Arduino sensors that could be used during the challenges, which helped them brainstorm ideas for possible attack vectors.

IV. CSAW Embedded Security Challenge (ESC)

For the CSAW ESC, the challenge mainly focused on acoustic side-channel attacks. In full, there were 6 challenges with flags and 2 open-ended challenges that needed to be captured and completed via acoustic side-channel analysis.

By the end of the competition, we had completed 4 out of 8 challenges:

- *Normal or Though* (Week 1, Challenge 1)
- *Friendly Disposition* (Week 1, Challenge 2)
- *KeyRing 1* (Week 2, Challenge 1)
- *Safecracker 1* (Week 4, Challenge 1)

A. Normal or Though

"Normal or Though" is an intriguing puzzle where the primary objective was to decipher and assemble a QR code from a sequence signals from the fan and servo motor. The problem context involved a peculiar interaction in a restaurant, leading to a realization linked to the QR code generated from the given Arduino's output.

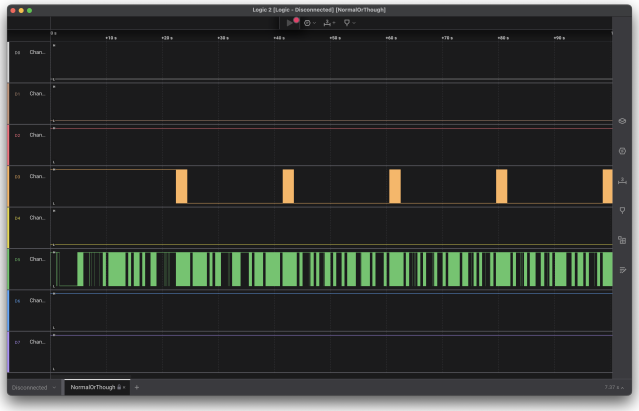

Fig. 1. The signals received by the logic analyzer. The servo motor signals are colored in orange, and the fan signals are colored in green.

This challenge was mainly completed by Tracy and Henry.

1) Setup and Tools Used: The tools used for this challenge primarily center around data analysis, measurement, and visualization software.

The tools the team used are:

- Audacity
- A logic analyzer and Saleae Logic 2
- Python (particularly, `Pillow` library)

The logic analyzer was used to capture the electronic signals from the fan and servo motors. Saleae's Logic 2 software was used to analyze the signals measured from the logic analyzer. The `Pillow` library in Python were used to manipulate and assemble image segments into a QR code format. Python was also used for assembling the QR code, particularly NumPy for matrix operations and Matplotlib for visualization.

2) Detailed Solution: The challenge had two parts. In the first part, the team was tasked with taking audio signals from a fan and servo motor and synthesizing them into a visible QR code. In the second part, the team was given two unlabeled matrices and had to determine which operations had to be applied to construct a second QR code.

The first part was completed by using signal analysis, while the second was completed using linear algebra knowledge.

Part 1: Signal Analysis and QR Code Formation:

In order to tackle the first part of the challenge, where the team was given inexplicable audio signals, the team started by recording the signals using a logic analyzer. This yielded a rather incomprehensible set of pulses coming from the fan and a consistent 33 pulses coming from the servo motor, as seen in Fig. 1.

In order to convert these signals into a QR code, the team hypothesized that stacking each of the segments (delineated by the servo motor signal) would yield a QR code. To test this hypothesis, the team (manually) took screenshots of each of the 33 segments ending at the end of a given servo motor pulse. The team then used the `Pillow` library in Python to

Fig. 2. QR Code Formation

stack these segments together and form a new image. The image that yielded was the QR code shown in Fig. 2.

Since the ESC focused on acoustic-side channel attacks, the team also considered an alternative acoustic approach for *Normal or Though*. In this alternative approach, the team would have created separate recordings of the servo motor and the fan, aligned the tracks on a common timeline in Audacity in order to visually, and audibly analyzed the interactions between the fan and the servo motor. The start and stop points of the servo motor, visible as distinct peaks or changes in the waveform, would denote the beginning and end of each QR code segment (similar to how a peak in the servo motor would indicate segments in the logic analyzer). With the servo motor segments identified, the corresponding segments of the fan's audio would represent the data for the QR code. Each segment would be precisely cut from the timeline where the servo motor starts and stops. Each audio segment corresponding to the fan's activity would be analyzed to extract binary data. This involves converting the audio waveform into a digital format, where specific sound characteristics (like frequency or amplitude changes) are translated into binary values. These binary values could then be assembled into a matrix form using Python's NumPy library. The assembled matrix represents the QR code, which could then be visualized using Matplotlib.

Part 2: Matrix Operations and Transformation: The QR code pointed to a GitHub page (https://gist.github.com/rostin79s/7c4ef6a4084c8f5e9b4060b8495ba532) which contained two matrices Q and R. Given the variable names, it was apparent that these matrices were components of a QR decomposition, which set the stage for the team's subsequent analyses.

Initial attempts to derive meaningful data involved performing operations such as $Q \cdot Q^T$ (dot product of Q and its transpose). The resulting matrix product ranged numerically between 0 and 25, prompting the team to try and encode these values as alphabetical characters. This approach did not yield any significant findings.

The team then proceeded with the product $Q \cdot R$, resulting in a 29x29 matrix. Considering the range of values from -2.205 to 2.023, the team experimented with creating a QR code by applying a threshold that turned values closer to 1 into '1's and all others into '0's. This attempt did not produce a functional QR code but hinted at the correct approach.

Fig. 3. Final Decoded QR Code

Final QR Code Generation: The team noticed that the Q and R in the code did not seem to align with the typical definitions of Q and R in QR decomposition. For typical QR decomposition, the following invariants are held:

- Q is an orthonormal matrix (i.e., $QQ^T = I$),
- R is an upper triangular matrix

What the team had instead observed was instead R was an orthonormal matrix and Q was a lower triangular matrix. Seeing this, the team decided to calculate $R \cdot Q^T$ and use the same threshold as the previous attempt, which was able to generate a QR code with the colors flipped. The breakthrough came from the following snippet of Python code, which correctly transformed the matrix data into a scannable QR code using the flipped threshold:

```
# Calculations
import numpy as np
import matplotlib.pyplot as plt

# R, Q from the gist
R = np.array([...])
Q = np.array([...])

QR = np.dot(R, Q.T)
abs_data = abs(QR)

# Threshold to convert values to binary
thresholded_data =
np.where(abs_data > 0.5, 0, 1)

# Plot the data to visualize the QR code
fig, ax = plt.subplots()
cax =
ax.matshow(thresholded_data, cmap='gray')
ax.axis('off')
plt.show()
```

This binary matrix was visualized to display a QR code (shown in Fig. 3), which pointed to a Google form: https://docs.google.com/forms/d/e/1FAIpQLSeRi2Q8ynUgFhZ_0_xmVqJdzWlKGXvWmEjAeLbfkSmWBRjaNg/viewform

The form provided the final challenge flag: **Kw1CkRe5p0Nze**, concluding the challenge with a successful

decode of the data.

It is important to note that the final decoded QR code does not appear to be perfectly formatted. However, it still successfully scanned. This is likely attributable to the built-in error correction capabilities of QR codes, specifically the Reed-Solomon error correction encoding.

B. Friendly Disposition

In "Friendly Disposition," competitors were given 4 trials (and a final meta-trial) where they were tasked with converting pulses of various timelengths and an encoding to ASCII characters. The goal was to determine the encoding, find the pattern, and use those encodings to find the flag.

This challenge was mostly completed by Henry.

1) Setup and Tools Used: The team used the following tools to solve the challenge:

- Audacity
- A logic analyzer and Saleae Logic 2
- Google Sheets (to organize the data)
- The Online Encyclopedia of Integer Sequences (OEIS)

The team used Audacity to measure the length of pulses, before transitioning to Saleae Logic 2 to get a more accurate length measurement. The team logged all of the data and organized it in Google Sheets and used OEIS to determine the patterns.

2) Detailed Solution: The entire Friendly Disposition challenge consisted of 5 parts (4 subchallenges and a final "meta"-challenge). Each had their own quirks, so they are explained here in separate sections.

Challenge 1: Uppercase: When accessing the first challenge, the team noticed that there were pulses that were only distinguished by their length. The team started by making a recording and measuring the lengths of the pulses in Audacity (Fig. 4).

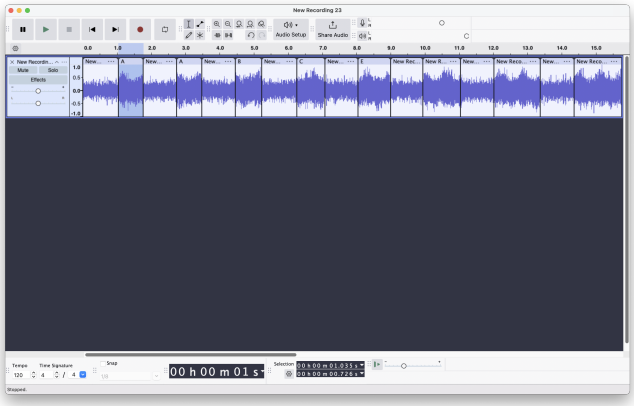

Fig. 4. Attempt in Audacity

While the length of each pulse could have been measured this way, the team decided to switch to a logic analyzer for more accurate measurements.

In recording the audio and measuring the results, the following data was produced:

Encoding	Signal Time (ms)
A	704
A	704
B	755
C	806
E	908
unknown	1061
unknown	1277
unknown	1724
unknown	1061
unknown	806
unknown	1214
unknown	1367
unknown	1928
unknown	1311
unknown	1265
unknown	1928

From the first 5 signals, the team noticed the difference between A and B, and B and C was around 50 and the difference between C and E was around 100, so the conversion was inferred to be:

1) Take the signal time s.
2) Compute $i = \frac{s-650}{50}$.
3) Convert number i to the closest capital letter (where $1 \to A$, $2 \to B$, $\ldots$, $26 \to Z$).

From there, the team computed the sequence to be $A, A, B, C, E, H, L/M, U/V, H, C, K, N, Y/Z, M, L, Y/Z$.

For L/M, U/V, and Y/Z, it was not entirely clear based on the pulse length measured. The team noticed the first six of the sequence (if you were to convert them back to numbers), were $1, 1, 2, 3, 5, 8$. From that, it was inferred that the pattern was Fibonacci mod 26, converted to capital letters, confirming the sequence:

Fibonacci	Fibonacci mod 26	Encoding
1	1	A
1	1	A
2	2	B
3	3	C
5	5	E
8	8	H
13	13	M
21	21	U
34	8	H
55	3	C
89	11	K
144	14	N
233	25	Y
377	13	M
610	12	L
987	25	Y

By continuing this sequence, the team obtained the keys K, J, U, E, Z, E, E, J.

Challenge 2: Numeric: Challenge 2 was a similar process. For challenge 2, the team ran into an issue where the first four

pulses provided were incorrectly all 650ms. This left only 1 pulse of actual information.

However, since the pulse's signal is repeated after a serial input is sent, so the team simply input $1, 2, 3, 4, 5, 6, 7, 8$ to capture the lengths and identify the signal encoding to be:

1) Take the signal time s.
2) Find the closest integer such that $i = \frac{s-650}{50}$.

From this process, the team obtained the encoding $2, 3, 4, 5, 7, 8, 9, 1, 3, 6, 7, 9, 3, 5, 7, 9$.

Seeing how challenge 1 took a pattern and applied modulo 26 to keep the encoding within A-Z, the team hypothesized that this challenge took a pattern and applied modulo 10 to keep the encoding within 0-9.

So, by undoing the modulo 10, the team found the sequence $2, 3, 4, 5, 7, 8, 9, 11, 13, 16, 17, 19, 23, 25, 27, 29$. Putting this sequence into OEIS, the team found that it matched the powers of primes (https://oeis.org/A000961).

From there, the remaining part of the sequence was inferred to be $31, 32, 37, 41, 43, 47, 49, 53, 59, 61, 64, 67, 71, 73, 79, 81$.

By applying modulo 10 to find the encodings, the team obtained: $1, 2, 7, 1, 3, 7, 9, 3, 9, 1, 4, 7, 1, 3, 9, 1$.

Challenge 3: Lowercase: For the third sequence, the same process applied. The team measured the signals, inferred the conversion to be:

1) Take the signal time s.
2) Compute $i = \frac{s-650}{50}$.
3) Convert number i to the closest lowercase letter (where $1 \to a, 2 \to b, \ldots, 26 \to z$).

From here, the team obtained $b, c, e, g, m, q, s, e, i, k, c, w, a, i, e, s$.

The first 7 of the sequence, when converted to numbers, yielded $2, 3, 5, 7, 13, 17, 19$. Putting this into OEIS resulted in the Mersenne exponents (https://oeis.org/A000043).

Extending this sequence, applying modulo 26, and converting to letters matched the remaining 9 encodings of the sequence. Thus, the pattern was continued, yielding s, s, o, c, q, i, g, u.

Challenge 4: Symbols: This challenge was strange in multiple ways. Immediately, the team noticed:

- Space was a symbol.
- The signal times were a lot longer than for the previous 3 challenges.

Since the signals were longer, the team inferred that the encoding was different. Based on the first 5 encodings measured (space, space, space, !, "), which are the first few characters of the printable ASCII character set (32 = space, 33 = !, 34 = "), the team inferred that space was the first character, ! was the second, and " was the third.

With this information, sequence $1, 1, 1, 2, 3$ was plugged into OEIS to find Narayana's cows sequence (https://oeis.org/A000930). This sequence made the most sense as the hint provided for challenge 4 referred to pulling a bull by its horns, which aligned well with the cow/cattle theme of the sequence.

The team compared this sequence with the first 9 measured pulses:

Narayana's cow sequence	Signal time
1	1612
1	1616
1	1616
2	1732
3	1847
4	1964
6	2194
9	2542
13	3004
...	...

Putting this table into Desmos and applying a linear regression yielded the linear equation $(\text{signal time}) = 115(\text{sequence value}) + 1500$. This linear regression had an R^2-value of almost exactly 1, indicating this conversion was likely correct and Narayana's cows sequence was almost certainly the sequence of this challenge. The full conversion therefore was:

- Take the signal time s.
- Compute $i = \frac{s-1500}{115}$.
- Convert number i to a symbol by finding the character with the ASCII code $(31 + i)$.

With this conversion, the team discovered the remaining sequence values to be $3, 12, 9, 12, 8, 1, 13$, which aligned with Narayana's cows mod 16.

With all the information in place, the team continued the Narayana's cows sequence, yielding sequence values $5, 6, 3, 8, 14, 1, 9, 7$. Converting this to symbols yielded `$ % " ' - space ( &`. This unfortunately did not work.

The team soon realized that the length of the pulse produced when *inputting* a symbol differed from the length of the pulse produced when *listening*. To remedy this, the team input `space ! " # $ % & '` to get the first 8 signals. The team found that the input symbol was off by 115ms (e.g., `space` was 1500ms instead of 1615ms, `!` was 1615ms instead of 1730ms, etc.).

Taking that in mind, the team converted the sequence numbers to pulse length and mapped that to which symbol would yield that pulse length. This resulted in the correct set of symbols, `% & # ( . ! ) '`.

Meta-Challenge: In the meta-challenge, the signals output used encodings from all of the 4 previous challenges, differentiated by the different variants of the motor sound (the sound of the motor in each of the 4 challenges differed, with the 4th challenge particularly having a loud, aggressive noise).

After recording the audio, the team measured the length of each pulse and which challenge each pulse's sound was most similar to. This yielded the following sequence:

Signal Time	Noise Type
1629	Uppercase
807	Numeric
1524	Lowercase
1728	Lowercase
791	Numeric
1364	Uppercase
806	Uppercase
800	Numeric
1496	Symbol
2426	Symbol
1371	Lowercase
647	Numeric
1673	Uppercase
1500	Symbol
1626	Lowercase
905	Uppercase
1520	Uppercase
1724	Uppercase
1612	Symbol
1371	Lowercase
1626	Lowercase
2539	Symbol

Using the mappings from the previous 4 challenges (and using the input version of the mapping for the symbol encoding), the team obtained: `S 3 q/r u 3 N C 3 space ( n/o 0 T/U space s/t E Q U/V ! n/o s/t )` (where `q/r`, `n/o`, `T/U`, `U/V`, etc. are ambiguous and could not be determined simply from the signals above).

Since the flag is probably readable text, the team picked the text `S3qu3NC3 (n0T sEQU!ns)`.

Most of the calculations were done on an Google Sheets spreadsheet.

C. KeyRing 1

For this challenge, competitors were given the .stl, .csv and .mp3 files of 4 labeled keys: A, B, C and D. The team was also given the .csv file or .mp3 file for 40 unlabeled sample keys. The task of the challenge were to identify these unlabeled samples as one of the 4 labeled keys (A, B, C, or D). The challenge also provided a minor hint that this information was obtained through an embedded microphone and vibration sensor.

This challenge was mostly completed by Shayan.

1) Setup and Tools Used: This challenge utilized Python and many Python libraries. The libraries used were NumPy, pandas, SciPy, fastDTW and librosa.

2) Detailed Solution: There are two ways to determine which key the unlabelled samples correspond to and it is either through the .mp3 file or the .csv file given for each sample.

The .mp3 files correspond to audio that was gathered from the 3D printer. In the samples given, the audio was much shorter than the full audio for each of the known labelled keys, meaning some time shifting had to be taken into account when analyzing and comparing the data.

The approach taken when considering the audio files was to first generate a spectrogram of each .mp3 file. The spectrograms represent the variance in frequencies over time of the audio files, which enabled the analysis of the .mp3 files in a more quantitative manner. These spectrograms were downscaled in practice as the large amount of data was too much for the machines used to handle. As mentioned before, the shorter time of recorded data for the samples and slight shifts in time for certain noises of the 3D printer had to be taken into account, leading the team to perform dynamic time warping when comparing known keys to the samples. Dynamic time warping takes into account these time shifts by warping time through timing offsets in order to minimize the sum of differences across the entire comparison. In order to properly perform the dynamic time warping, the spectrograms needed to be "flattened" into one dimension so that the Python fastdtw library function can operate on the data. A comparison metric also needs to be specified and euclidean distance was used, as the data was already flattened into one dimension. Once the conditions to run fastdtw were met, fastdtw was applied for each sample against each of the four known keys. The minimum sum of distances returned by the fastdtw function of the four key comparisons was printed as the most likely key for that sample for the team's solution.

Audio solutions: Sample 3: C, Sample 4: C, Sample 5: C, Sample 6: D, Sample 7: D, Sample 8: D, Sample 10: D, Sample 13: C, Sample 14: C, Sample 17: C, Sample 19: C, Sample 20: C, Sample 24: D, Sample 25: C, Sample 26: C, Sample 29: C, Sample 35: C, Sample 36: D, Sample 37: C, Sample 40: C.

The .csv files correspond to vibration sensor data gathered from the 3D printer, which typically corresponds to the acceleration of the nozzle. They contained the values in the X, Y, and Z direction at each timestamp (which incremented by 2). The X and Y directions were the point of focus in the team's solution as they should correspond to the outline of the keys and more specifically the teeth. The approach given the X and Y data was to perform Fourier Fast Transforms on each key's X and Y data. FFTs have been used in similar situations to classify objects as FFTs give values in the frequency domain rather than time domain. This enables the analysis of a spectrum of frequency in the X and Y directions of the vibration sensor data that should would better help match keys (as keys with the same teeth will have similar frequencies and frequency amplitudes). Once generating FFTs for the X and Y data for each key and sample, the X and Y FFT data was combined by taking the euclidean norm of both. From here, fastdtw was applied again and although time was not a factor when looking at the combined FFT data, fastdtw should still take into account discrepancies in the amount of data provided between each sample and all the known keys as well as slight shifts in frequencies. The minimum distance returned by the dynamic time warping performed on each sample against all known keys corresponded to the key the sample best fit for the team's solution.

Vibration data solutions: Sample 1: D, Sample 2: B, Sample

9: D, Sample 11: B, Sample 12: C, Sample 15: B, Sample 16: B, Sample 18: D, Sample 21: C, Sample 22: B, Sample 23: B, Sample 27: B, Sample 28: D, Sample 30: B, Sample 31: B, Sample 32: D, Sample 33: C, Sample 34: B, Sample 38: B, Sample 39: B.

D. Safecracker 1

In this challenge, competitors are given the source code for a lock (with the keys hidden) and a recording of the code in action and are tasked with determining the number combination for the lock.

This challenge was mostly completed by Henry.

1) Setup and Tools Used: The team used Audacity (to measure the length of pulses) and a calculator.

2) Detailed Solution: Looking at the source code, the team found that all of Safecracker 1's entries are at default speed using the single motor mode.

The team also found in the source code that the demo recording runs a motor in all of the modes, both forwards and backwards, for 100 steps.

Thus, the team measured the single forwards and backwards pulses to determine how much a step is. The team found that 100 single steps took around 3.2s to complete (so 1 step is approximately 32ms).

The team then measured the length of the pulses from the recording of the SafeCracker 1 recording.

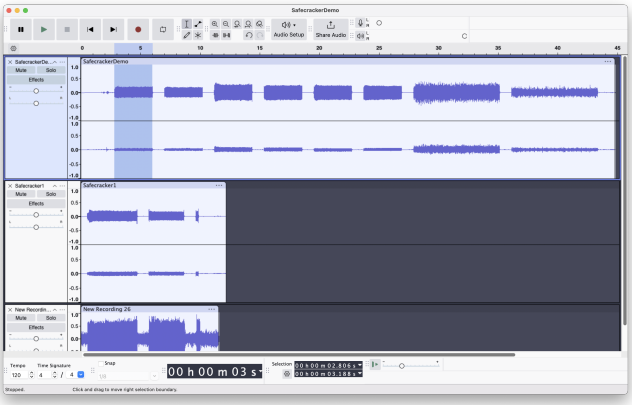

Fig. 5. Another picture of Audacity, used to measure the pulse lengths from the demo

From these recordings, the pulses were found to take approximately 4.2s, 3s, and 0.238s (131 steps, 93 or 94 steps, and 7 or 8 steps). This ultimately was not correct.

The team then continued to refine this result, simply by zooming in in Audacity and making more precise selections. This resulted in pulses of 4.2s, 2.946s, and 0.238s (131 steps, 92 steps, and 7 steps). This ended up being the correct answer.

V. Individual Contributions

A. Shayan Aqeel

Initially, Shayan Aqeel's role on the team had been assisting in the research of side-channel attacks on cyber physical systems geared towards the CSAW competition. He had accomplished this by including a section on attack methodologies in the preliminary report and highlighting potential procedures to exploit the ATmega328P chip for the Arduino Uno OSINT paper.

Within the preliminary report, Shayan noted the importance of a proper understanding of the system and ways to collect relevant hardware data. He also explained how to begin testing attack techniques by utilizing the data collected and through fault injection analysis. In the OSINT paper, he emphasized key aspects of the ATmega328P chip found on the Arduino Uno R3 boards which could provide use in the upcoming challenges. Extracting firmware, changing the frequency of clock cycles and obtaining precise timing information were among the considered features.

Going into the competition, he helped set up the hardware along with the rest of the team and successfully ran the hardware testing binary on a Windows machine. His main focus in the competition was working on week 2 challenge 1, Key Ring 1. The process for obtaining the team's solution has been mentioned previously but Shayan's overall approach involved much more trial and error until deciding on a final procedure due to the more open-ended nature of the challenge. Many different combinations of data normalization techniques and time warping effects were tested along with visualizations of the solutions to select the most accurate and complete solution technique. After finishing the challenge and preparing the competition deliverables such as the final paper, poster and presentation with the rest of the team, Shayan began looking into week 3 challenge 2, Fast and Max. He was able to run both the dummy and real binaries on the hardware as well as begin reverse engineering both binaries through Ghidra but eventually hit a dead end as the semester wrapped up.

B. Henry Bui

Henry Bui's focus has mainly been towards research of current side-channel attack methods (both towards general cyber-physical systems and towards manufacturing-oriented cyber-physical systems), performing formatting and revisions of reports completed for the competition, and researching possible augmentations to the Arduino Uno R3 that could be included for this year's ESC.

For the preliminary report, Henry discussed the implementation of an acoustic side-channel attack on manufacturing cyber-physical systems (from the work performed by Al Faruque et al.). He also discussed how certain types of side-channel attacks—specifically timing analysis and differential power analysis—are used to access private keys of cyber-physical systems. He also read through the suggestions provided by the team's advisor and edited the paper in accordance to those suggestions. Finally, Henry also collected all of the references used within the paper and set up the `bibtex` package to properly manage the bibliography.

For the Arduino OSINT, Henry added onto Tracy's discussion of the Arduino Uno R3 and its EEPROM by discussing a possible exploit towards the EEPROM. Additionally, Henry

discussed the devices attached to the Arduino Uno R3 from the 2023 CSAW ESC (such as the 7-segment display and microphone), based on the information surrounding the 2023 competition provided by the organizers.

For the CSAW ESC competition, Henry's contributions mainly involved the progress for 3 of the challenges: Normal or Though, Friendly Disposition, and Safecracker 1.

In Normal or Though, he performed the recording of electric signals sent to the fans and servo motor through the logic analyzer, broke up the signals into the 33 segments, and constructed it into a QR code with Python. He also proposed the fix to QR decomposition in the second part of the challenge.

In Friendly Disposition, he did all of the technical work to complete the challenge. He recorded the signals through the logic analyzer, did the measurements, sequence identification, and solving of the meta challenge.

Finally, for Safecracker 1, he looked through the source code to understand what to find and measured the length of the pulses through Audacity.

He also traveled to New York to compete in the CSAW ESC competition in person. Here, he and Smit presented the team's progress to the judges and the other competitors in the challenge through a live presentation and through a poster panel. In the presentation and poster panel, the two discussed the technical aspects to completing each challenge, talked to the judges about their thought processes, and attended the award ceremony where the competition judges announced that Yellow Hackets won first place!

C. Tracy Guo

Tracy Guo served as the team lead for the CSAW ESC Yellow Hackets team, managing tasks such as registering the team for the competition, coordinating team meetings, communicating with organizers, and keeping team members updated on deadlines and requirements. She facilitated weekly subteam meetings to assign tasks, discuss progress on challenges, and ensure alignment on deliverables, including the preliminary and final reports, presentations, and the competition poster.

During the CSAW ESC competition, Tracy contributed to several technical aspects of the project. She led the OSINT research on the Arduino Uno R3 microcontroller, focusing on its architecture and EEPROM functionality. This research supported the team's work on hardware challenges and informed the technical approach for solving competition problems. Tracy also worked on Week 1 Challenge 1, analyzing data from a logic analyzer to decode fan and servo motor signals into a QR code. She applied techniques to map clock cycles and segment signals with the collaboration of Henry, as well as developing scripts to process the data to generate the final QR code.

Tracy created a plan for Week 3 challenges, which included analyzing motor signals and identifying patterns in movement data for decoding. The plan outlined steps for recording and processing hardware outputs and applying semaphore cipher to interpret the encoded information.

Tracy wrote sections of the preliminary report, focusing on mitigation strategies for side-channel attacks. She also developed templates for the team's poster and final report, contributed sections on solved challenges, and integrated feedback from teammates and advisors. Tracy was responsible for preparing and testing the team's presentation, ensuring that all materials, including audio and slides, were functional during class meetings.

In addition to technical contributions, Tracy set up the Arduino hardware and software in the lab, documented troubleshooting steps, and recorded data for challenges. She also managed logistical issues, such as resolving team member registration confirmations and coordinating hardware deliveries. After the competition, she participated in finalizing the remaining challenges, writing additional sections of the final paper, and preparing for the final class presentation and peer evaluations.

D. Smit Patel

For this class, Smit Patel has been diving deep into embedded cybersecurity, with a special focus on side-channel attacks. With his background in ICS (Industrial Control Systems) and OT (Operational Technology) protocols, he has a solid understanding of how manufacturing systems operate. However, he recognized that he needed to expand his knowledge in side-channel attacks and embedded security to grow further in this field.

During the first 8-weeks of the course, Smit was learning from various sources, such as watching YouTube tutorials on embedded security and digging into research papers on side-channel attacks. He's also taking an IoT hardware hacking course with TCM Security to gain hands-on experience with hardware vulnerabilities. For his preliminary qualification report, Smit explored different types of side-channel attacks, from timing and energy consumption analysis to electromagnetic and acoustic emission techniques. In addition, he conducted some open-source intelligence (OSINT) research on an Arduino PLC starter kit. This research could prove valuable for upcoming challenges, as the Arduino platform might be useful in managing the manufacturing plant's operations.

During the second 8-weeks of the course, Smit helped review the presentations for the in-person event. He also, presented two of the challenges to the judges at the CSAW event. During this time he has helped them team with class presentations needs as well. Furthermore, he has enjoyed this new side of cybersecurity; hence, embedded security and how fun it is. Therefore, due to restrictions of not being in-person he has spent a lot of his time learning hardware hacking from TCM Security and YouTube and plans to take a certification exam along with getting couple of IoT devices to pentest and further develop his skill set this winter break.

VI. Conclusion

Overall, a solid understanding of the Arduino Uno R3 and side-channel attack methodologies and mitigations of cyber-physical systems provided a strong foundation going into this

year's CSAW competition. Data collection techniques, known attack approaches and recognized hardware vulnerabilities helped give direction for the weekly challenges. With this knowledge and a mixture of software and hardware analysis skills, the Yellow Hackets team was able to provide solutions for the Normal or Though, Friendly Disposition, Key Ring 1 and Safecracker 1 challenges. These solutions along with the in-person poster and presentation components of the competition were enough to secure first place in the competition for the North America region. Following the competition, the team worked on the other challenges with references to other teams' and the organizer's solutions in order to address weaknesses and gain embedded systems knowledge for upcoming semesters.

REFERENCES

[1] J. Fan, X. Guo, E. De Mulder, P. Schaumont, B. Preneel, and I. Verbauwhede, "State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures," in *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, Jun. 2010, pp. 76–87. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5513110

[2] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Advances in Cryptology — CRYPTO' 99*, M. Wiener, Ed. Berlin, Heidelberg: Springer, 1999, pp. 388–397.

[3] M. A. Al Faruque, S. R. Chhetri, A. Canedo, and J. Wan, "Acoustic Side-Channel Attacks on Additive Manufacturing Systems," in *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*, Apr. 2016, pp. 1–10. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7479068

[4] M. A. Khair, J. R. P. K. Ande, D. R. Goda, and S. R. Yerram, "Secure VLSI Design: Countermeasures against Hardware Trojans and Side-Channel Attacks," *Engineering International*, vol. 7, no. 2, pp. 147–160, Dec. 2019. [Online]. Available: https://abc.us.org/ojs/index.php/ei/article/view/699

[5] C. Liptak, S. Mal-Sarkar, and S. A. Kumar, "Power Analysis Side Channel Attacks and Countermeasures for the Internet of Things," in *2022 IEEE Physical Assurance and Inspection of Electronics (PAINE)*, Oct. 2022, pp. 1–7. [Online]. Available: https://ieeexplore.ieee.org/document/10014854

[6] Y. Li, M. Chen, and J. Wang, "Introduction to side-channel attacks and fault attacks," in *2016 Asia-Pacific International Symposium on Electromagnetic Compatibility (APEMC)*, vol. 01, May 2016, pp. 573–575. [Online]. Available: https://ieeexplore.ieee.org/document/7522801

[7] A. Johnson, "Side Channel Attacks and Mitigations 2015-2020: A Taxonomy of Published Work," Jun. 2021. [Online]. Available: https://www.proquest.com/conference-papers-proceedings/side-channel-attacks-mitigations-2015-2020/docview/2555179734/se-2