Final Project Report

Introduction to Hardware Security

Co-author:

111000225 111000212 111000178
張皓翔 吳承翰 連正文

Introduction

Concept Overview

Initially, we aimed to identify a source of random events. Common examples of random events, as discussed in lectures, include thermal noise, radio frequency noise, and clock jitter noise… However, during our brainstorming process, we wondered: why not use something more ubiquitous in our daily lives—a common sensor? Specifically, an external sensor that generates noise events. Inspired by the approach of the established cybersecurity company Cloudflare, which uses an entire wall of lava lamps as a random seed for SSL/TLS encryption, we were intrigued by their observation:

"The 'real world' turns out to be a great source for randomness, because events in the physical world are unpredictable.

This led us to consider one of the most common, high-information sensors: **cameras**. From mobile phones in our hands to street cameras, satellites in orbit, and even Tesla vehicles, more and more devices, including mobile devices, are being equipped with cameras. With recent advancements in CMOS technology and the reduction in manufacturing costs, the image sensor market continues to grow.

What types of noise can we expect from using image sensors?

- Environmental changes
- Human appearance
- Light and shadow fluctuations
- Resolution changes
- Lens distortion
- Encoding format variations

Where the Randomness Come From

We observed that IMU sensors generate a significant amount of random noise in their data. This means that even when the IMU remains stationary, the collected data exhibits noticeable jitter or fluctuations. To leverage this characteristic, we propose using the inherent data jitter from the IMU as our entropy source for random number generation.

Such an approach takes advantage of the stochastic nature of IMU sensor noise, which arises from environmental factors, electronic interference, and inherent imperfections in the sensor itself. By carefully processing this noise, we aim to extract high-quality entropy, which can be utilized in cryptographic applications or simulations requiring a robust and unpredictable random number source. This method also aligns well with the principles of hardware-based randomness, where physical phenomena are harnessed for entropy generation, offering advantages in unpredictability compared to algorithmic methods.

In addition to the IMU sensor, we also utilized a camera as another source of randomness. Cameras, as widely used sensors in various applications, inherently capture data with some degree of noise. This noise originates from factors such as sensor imperfections, thermal effects, and environmental conditions like lighting fluctuations. Even in a controlled, static environment, the captured images contain pixel-level variations that can serve as a source of entropy.

Moreover, leveraging cameras as entropy sources complements the IMU-based approach. While the IMU provides randomness rooted in motion and jitter, the camera contributes randomness based on visual data variability. Together, these sensors form a robust system for generating unpredictable random numbers suitable for cryptographic purposes, secure communications, or complex simulations. Future work will focus on refining data processing techniques to maximize entropy extraction while minimizing deterministic patterns in the noise.

Experimental Environment Setup

Experiment Implementation

In this experiment, we utilize both the **camera** and **IMU** from mobile devices. The specific methods of leveraging these components will be detailed in the following section. For the camera, we directly use the built-in webcam of a laptop. To further enhance randomness, we activate the computer's "automatic framing" feature, which allows the camera to move dynamically—a function increasingly supported by many mobile devices through software.

Camera Setup

- **Device**: Built-in laptop webcam.
- **Enhancement**: Enabled "automatic framing," adding variability by introducing autonomous camera movements.
- **Rationale**: Cameras are ubiquitous in mobile devices, making this approach highly replicable and cost-effective.

IMU Setup

The IMU required more effort to integrate due to concerns about insufficient entropy in Plan A. To address this, we added an extra source as a backup option.

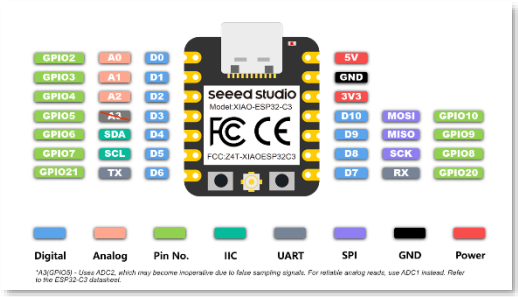
1. **Deployment Concept**

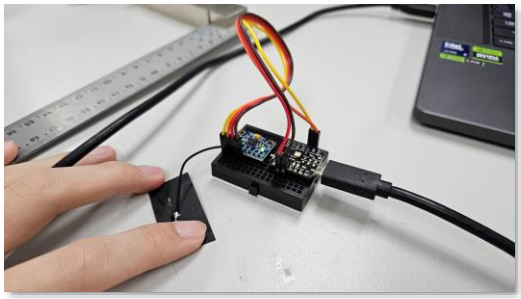
   IMUs are assumed to be worn by users. Like cameras, IMUs are now standard in wearable devices, particularly accelerometers. They are low-cost and widely available in smartwatches, rings, and other wearable technologies. We aimed for a wireless sensor to collect human acceleration data conveniently.

2. **Hardware Selection**

   We chose the XIAO ESP32-C3 development board for its advantages:

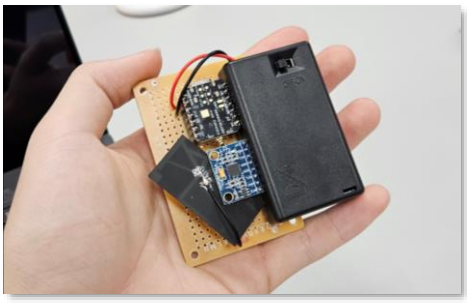
   - **Dual-core processor** with high clock speed.
   - **2.4 GHz wireless communication** capability.
   - **Compact size**, enabling easy attachment to the user's body.

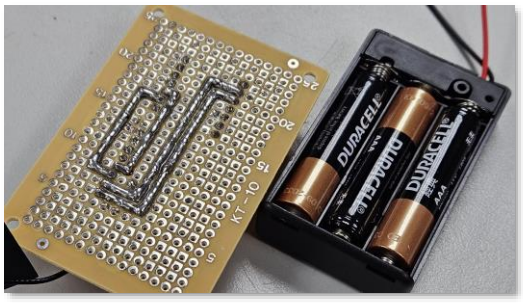




3. **Circuit Modifications**

   - A simple circuit board was soldered to enable battery-only power supply for portability.

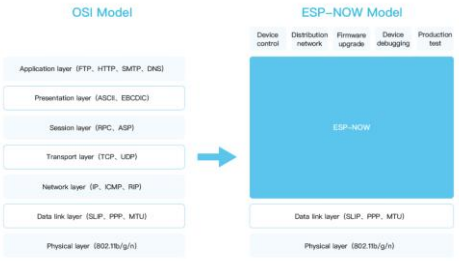




4. **Communication Protocol**

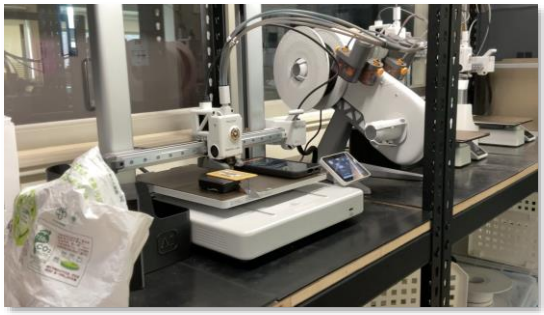
   Instead of using Wi-Fi or Bluetooth, we adopted the specialized "**ESP-NOW**" protocol, which:

- Simplifies the upper five layers of the OSI model, reducing handshake times.
- Provides fast response, **low latency**, and long-range communication.
- Supports **mesh** functionality, allowing multiple IMUs or additional future sensors to be interconnected for greater **robustness and scalability**.



Validation and Comparison

To compare the noise differences across sensors, we used a 3D printer as the **ground truth** for controlled motion. The printer followed specific g-code instructions to perform reciprocating movements at varying speeds.

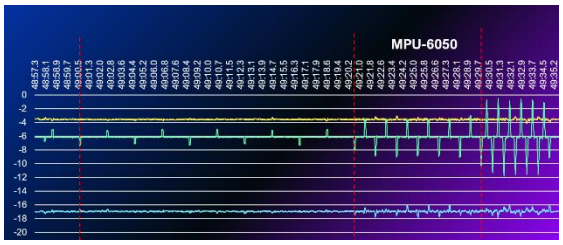


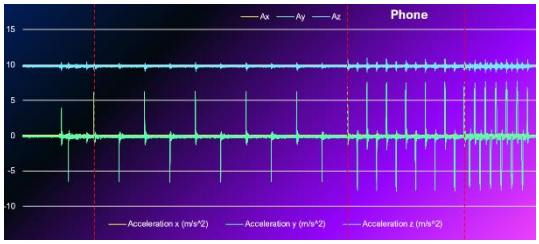
```
IMU-02 > IMU_test.gcode
1  M17 X1.2 Y1.2 Z0.75
2  G90
3  M83
4  G28 ; home all axis
5  G1 X128 Y128 Z1 ; move to center
6  G29.2 S0 ; turn off bed leveling compensation
7
8  ; Move Z axis to 128mm
9  G1 Z128
10
11 ; Start Y-axis oscillation - 10 times at 3000 mm/min
12 G1 X128 Y100 F3000 ; move to start position
13 G1 X128 Y200 F3000 ; move to end position
14
15 G1 X128 Y100 F3000 ; move back to start
16 G1 X128 Y200 F3000 ; repeat 1
```

Observations

- Comparing the MPU-6050 with the sensors in a smartphone revealed significant differences in randomness, even when subjected to identical movements. These differences are influenced by the sensor's characteristics, configuration, and even environmental factors like **temperature and humidity.**

- The MPU-6050 communicates via the **I2C** protocol, enabling multiple slave devices. With two IMUs available, we simultaneously collected data from both, maximizing entropy.

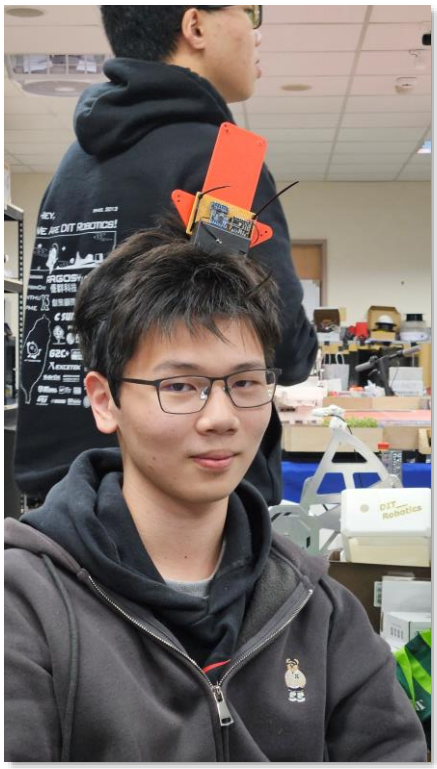




Mounting Solution

For practicality and aesthetics, we designed a **3D-printed headband** to securely attach the IMUs during testing.

This approach not only demonstrates variability in noise levels across different sensors but also provides a robust setup for future experimentation and scalability.



Presentation Design

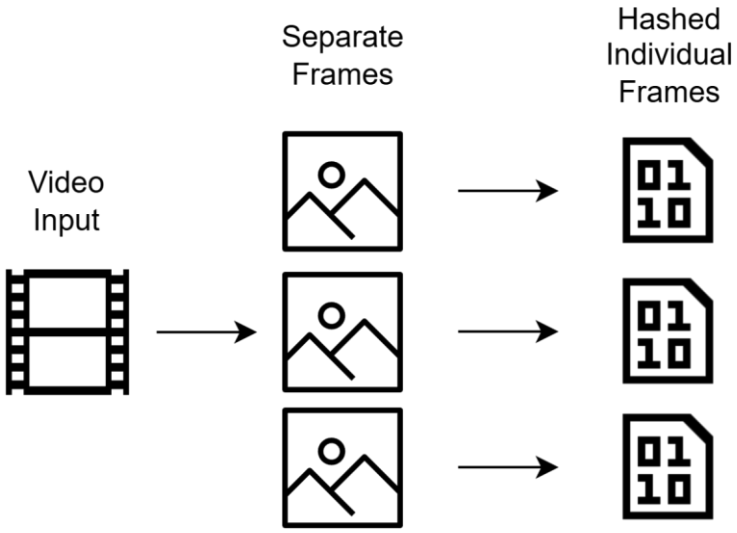
Initial Design Overview

Our initial design attempted to create a **True Random Number Generator (TRNG)** by combining video input processing with IMU sensor data. The system was designed to be a continuous generation process that would produce random bits through iterative hashing. The design was thought of by utilizing the unpredictable nature of video input combined with motion sensor data to create a reliable source of randomness.

Video Frame Processing

The first stage of our design focused on gathering entropy from video input, operating under the assumption that video frames would contain sufficient environmental randomness.

1. Video Capture
    - Video input capture from a camera source for a set amount of seconds
    - Constant frame rate capture
    - **Raw frame data stored in RGB format**
2. Frame Extraction
    - Individual frame isolation from the video stream
    - Each frame processed as a distinct entropy source
    - Preservation of full frame resolution to maximize potential entropy
3. Frame Hashing
    - Application of SHA-256 hash function to each frame's raw data
    - Generation of 256-bit hash output per frame
    - Each hash is a unique entropy contribution

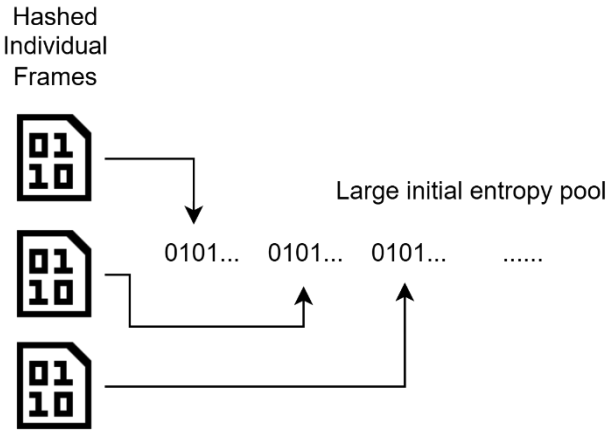


Hash Concatenation Process

The next stage involved creating a larger entropy pool, which we believed would increase the randomness quality and allow for continuous generation of random bits.

Initial Concatenation

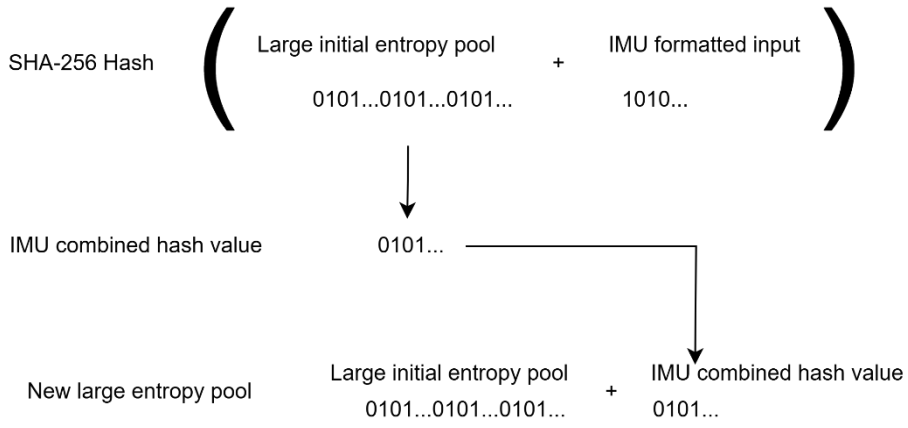
- Sequential combination of frame hashes
- Each 256-bit hash appended to the growing string
- Storage of complete hash strings without truncation



Continuous Generation Mechanism

To maintain continuous operation and introduce additional entropy, the system was designed with a perpetual generation loop:

1. IMU Data Integration
    - Regular sampling of IMU sensor data
    - Collection of acceleration and gyroscope readings
    - Formatting of sensor data for hash input
2. Concatenation Process
    - Addition of IMU data to existing hash string
    - Preservation of temporal relationship between readings
    - Maintenance of data format consistency
3. Hash Generation Loop
    - Application of SHA-256 to combined string
    - Generation of new 256-bit hash values
    - Storage of results in the continuing sequence



Binary Output Generation

The final stage implemented the conversion of hash values into usable random numbers:

Hash Processing

- Sequential processing of generated hash values
- Conversion of hexadecimal to binary representation
- Preservation of full 256-bit sequences

Implementation Considerations

During the design phase, several implementation details were considered:

Selection of SHA-256

- Wide availability in standard cryptographic libraries
- Fixed output size of 256 bits, suitable for our binary conversion needs
- Fast computation speed compared to other cryptographic hash functions
- Avalanche effect, different image frames with little changes will output completely different hash values
- Uniform distribution, output bits are distributed uniformly in cases of common pixel values.

The above design represented our initial understanding of TRNG requirements and attempted to create a system by combining multiple entropy sources. However, as will be discussed in subsequent sections, this approach contained several fundamental flaws that were identified during the presentation phase. These issues primarily centered around misunderstandings of entropy principles and hash function properties.

Design Analysis

Before the presentation where design flaws were identified, we conducted extensive testing on our TRNG implementation to evaluate its statistical properties. While these tests showed promising results at first glance, they ultimately did not validate the fundamental soundness of our design.

IID Test Results

We performed a series of statistical tests assuming the sequence was Independent and Identically Distributed (IID):

1. Basic Statistical Measures
    - Mean: 0.499859 (ideal: 0.5)
    - Median: 0.5 (ideal: 0.5)
    - Original Entropy (H): 0.995921 (ideal: 1.0)
2. Chi-Square Analysis
    - Independence Test p-value: 0.499849
    - Goodness of Fit Test p-value: 0.482012
3. Test Case Pass Rate: 87.65%

```
Opening file: '../formatted_binary_hash_chain.bin' (SHA-256 hash 043951cba2fb0c5881b0d4a9753cad
Loaded 1020928 samples of 2 distinct 1-bit-wide symbols
Calculating baseline statistics...
        Raw Mean: 0.499859
        Median: 0.500000
        Binary: true

Literal MCV Estimate: mode = 510608, p-hat = 0.50014104814443328, p_u = 0.50141569453893486
H_original: 0.995921
Chi square independence
        score = 2045.357622
        degrees of freedom = 2046
        p-value = 0.499849

Chi square goodness of fit
        score = 8.527172
        degrees of freedom = 9
        p-value = 0.482012

** Passed chi square tests

Literal Longest Repeated Substring results
        P_col: 0.500000
        Length of LRS: 43
        Pr(X >= 1): 0.057522
** Passed length of longest repeated substring test
```

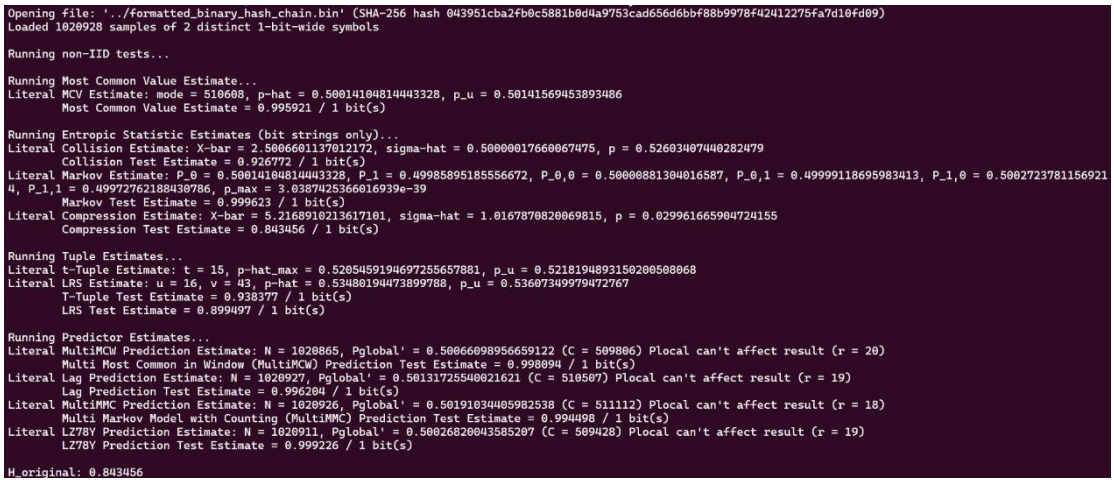
```
Beginning permutation tests... these may take some time
 87.65% of Permutation test rounds, 100.00% of Permutation tests

                statistic  C[i][0]  C[i][1]  C[i][2]
-------------------------------------------------------
                excursion        6        0       15
       numDirectionalRuns        6        0       42
       lenDirectionalRuns        4        6        0
    numIncreasesDecreases        6        0        9
            numRunsMedian        6        0        7
            lenRunsMedian        7        4        2
             avgCollision        6        0        7
             maxCollision        5        1        5
           periodicity(1)       10        0        6
           periodicity(2)       17        0        6
           periodicity(8)      102        0        6
          periodicity(16)       16        0        6
          periodicity(32)        6        0        6
            covariance(1)        8        0        6
            covariance(2)        6        0       29
            covariance(8)        6        0        6
           covariance(16)        6        0       12
           covariance(32)        6        0        6
              compression        6        0       14
(* denotes failed test)
```

Non-IID Test Results

Additional tests were performed without the IID assumption:

1. Most Common Value (MCV) Test
    - p-hat: 0.500142 (ideal: 0.5)
2. Collision Test
    - x-bar: 2.500066
3. Entropy Analysis
    - H_original: 0.84456



Test Result Analysis

While these statistical results initially appeared promising:

- Most metrics were close to their ideal values
- High pass rate in standard test cases
- Chi-square test p-values suggested randomness
- Entropy measurements were within acceptable ranges

However, as we later discovered during the presentation, passing these statistical tests does not validate the fundamental soundness of a TRNG design. The following section details the critical flaws in our approach that were identified despite these seemingly positive test results.

Input-Output Entropy Violation

The most significant flaw in our design was the violation of a fundamental principle in random number generation: the output entropy cannot exceed the input entropy. Our design made several critical mistakes:

- Concatenating multiple SHA-256 hashes does not increase the actual entropy
- Each frame's entropy is limited by its source randomness, not the hash output size
- The total entropy cannot exceed the sum of actual input entropy from video frames and IMU data

This analysis of our initial design's flaws proved valuable in understanding the complexity of TRNG design and the importance of following the fundamental cryptographic principles.

Post-Presentation Improvement

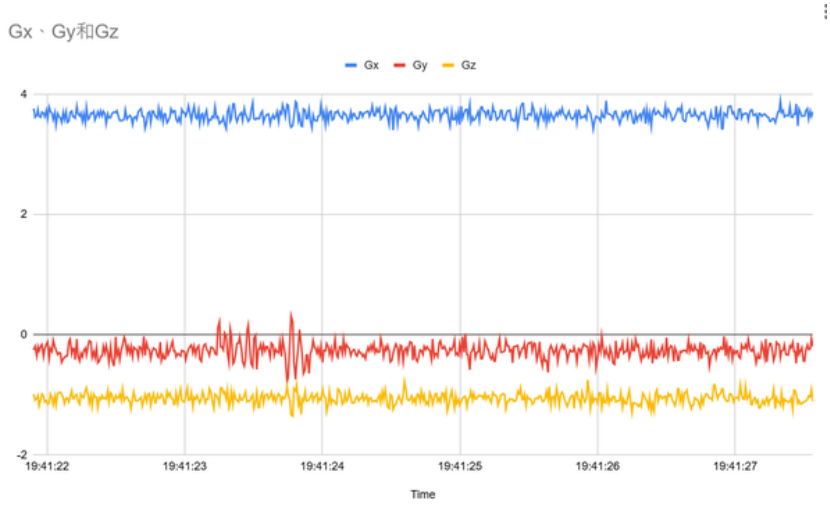
Thanks to the teacher's suggestions, we realized that our process did not align with the NIST standards or the definition of an entropy source. Specifically, we identified two key issues in our original methodology:

1. **Inappropriate Testing Workflow:** According to the guidelines, tests for IID (Independent and Identically Distributed) and non-IID properties must not be conducted after applying a hash function. Doing so could obscure the true statistical properties of the original data and lead to misleading conclusions.
2. **Misinterpretation of Entropy Compression:** Random number generation must only compress the entropy of the source and not expand it. In our initial approach, we inadvertently generated multiple bits from a smaller number of bits. This practice aligns more closely with pseudo-random number generation

(PRNG) rather than a true entropy source.

After identifying these issues during the presentation, we redesigned and conducted our experiments. To ensure compliance with the standards, we focused solely on using raw data from the IMU sensor without applying any transformations that could distort the data's inherent randomness. By adhering to this stricter methodology, we aim to demonstrate that the IMU data itself can serve as a valid entropy source while ensuring our process aligns with the principles outlined by NIST.

IMU digitalization method



Jitter of IMU sensor

Our new entropy source is derived from the natural jitter inherent in the IMU itself. Even in a stationary state, the IMU exhibits fluctuations in both acceleration and angular velocity, providing a basis for generating randomness.

To address the significant bias often observed in acceleration measurements—such as the +1G gravitational acceleration bias in certain directions—we opted not to use the absolute acceleration values. Instead, we focused on **relative changes** in acceleration. Specifically, when the acceleration increases, we output a "1," and when it decreases, we output a "0." This simple yet effective approach allows us to generate a continuous stream of random bits based on the IMU's inherent noise. To balance the number of “1” and “0”, we apply xor to neighbor value.

The primary advantage of this method is that all generated random numbers are derived directly from the IMU's noise, which originates from natural microscopic variations and the sensor's internal imperfections. This ensures that the randomness is rooted in physical phenomena, making it a robust source of entropy.

However, a notable limitation of this approach is its relatively low throughput. The IMU provides only 250 samples per second, which constrains the rate at which random bits can be generated. Despite this, the trade-off between throughput and the quality of randomness makes this method well-suited for applications where high-quality entropy is more critical than generation speed. Future work could explore combining data from multiple IMUs or optimizing sampling techniques to increase throughput while maintaining randomness quality.

The accurate throughput of our post-presentation method is directly tied to the sampling rate of our IMU sensor. By measuring the elapsed time, we determined that the actual sampling rate of the IMU is approximately **248.7 Hz**. Additionally, each sample provides six data points: three-dimensional acceleration and three-dimensional gyroscope measurements. This means the theoretical maximum throughput for raw data is **6 × 248.7 = 1,492.2 data points per second**.

Non-IID Test Results

```
Opening file: '../reshaped_IMU_data.bin' (SHA-256 hash 764f2b275f84e573c071ae69b75788e609758b13f7d0f0f50ffa9dff024d4c42)
Loaded 1048576 samples of 2 distinct 1-bit-wide symbols

Running non-IID tests...

Running Most Common Value Estimate...
Literal MCV Estimate: mode = 524639, p-hat = 0.50033473968505859, p_u = 0.50159246915505562
        Most Common Value Estimate = 0.995412 / 1 bit(s)

Running Entropic Statistic Estimates (bit strings only)...
Literal Collision Estimate: X-bar = 2.4999916562029733, sigma-hat = 0.50000058906531231, p = 0.53218740669875597
        Collision Test Estimate = 0.909994 / 1 bit(s)
Literal Markov Estimate: P_0 = 0.50033473968505859, P_1 = 0.49966526031494141, P_0,0 = 0.50088823150438966, P_0,1 = 0.49911176849561034, P_1,0 = 0.4997795536486257
, P_1,1 = 0.50022044635513743, p_max = 3.6842221392853686e-39
        Markov Test Estimate = 0.997452 / 1 bit(s)
Literal Compression Estimate: X-bar = 5.2149974422015193, sigma-hat = 1.0170953744355935, p = 0.0318759115054306 76
        Compression Test Estimate = 0.828565 / 1 bit(s)

Running Tuple Estimates...
Literal t-Tuple Estimate: t = 16, p-hat_max = 0.52689672779428934994 41, p_u = 0.52815263645890925524 88
Literal LRS Estimate: u = 17, v = 36, p-hat = 0.50004257252255196, p_u = 0.50130302269848 81
        T-Tuple Test Estimate = 0.920973 / 1 bit(s)
        LRS Test Estimate = 0.996253 / 1 bit(s)

Running Predictor Estimates...
Literal MultiMCW Prediction Estimate: N = 1048513, Pglobal' = 0.50152147415417592 (C = 524533) Plocal can't affect result (r = 23)
        Multi Most Common in Window (MultiMCW) Prediction Test Estimate = 0.995617 / 1 bit(s)
Literal Lag Prediction Estimate: N = 1048575, Pglobal' = 0.50105602793897241 (C = 524076) Plocal can't affect result (r = 18)
        Lag Prediction Test Estimate = 0.996956 / 1 bit(s)
Literal MultiMMC Prediction Estimate: N = 1048574, Pglobal' = 0.50170118985960255 (C = 524752) Plocal can't affect result (r = 20)
        Multi Markov Model with Counting (MultiMMC) Prediction Test Estimate = 0.995100 / 1 bit(s)
Literal LZ78Y Prediction Estimate: N = 1048559, Pglobal' = 0.50133737302795522 (C = 524363) Plocal can't affect result (r = 18)
        LZ78Y Prediction Test Estimate = 0.996146 / 1 bit(s)

H_original: 0.828565
```

Additional tests were performed without the IID assumption:

1. Most Common Value (MCV) Test
    - p-hat: 0.5003347 (ideal: 0.5)
2. Entropy Analysis
    - H_original: 0.828565

Test Result Analysis

The entropy obtained using the non-IID test in our post-presentation method is slightly lower than that of our original approach. This indicates that the randomness in the data has decreased marginally. Furthermore, the method does not pass the IID test, suggesting that the data lacks the statistical properties required for independence and identical distribution. This result highlights a trade-off between adhering to entropy source principles and achieving high-quality randomness. While the method ensures compliance with entropy generation standards by relying solely on raw IMU data, it underscores the challenge of balancing theoretical correctness with practical performance in randomness generation

Final Thoughts

Great course! Great teacher! Great TA <3 muah.

[The Handsome Lecturer at the Job Fair 👇]

[張皓翔]

In this final project, we gained our first hands-on experience in implementing a True Random Number Generator (TRNG). However, due to the intense workload of this 16-week course, overlapping with multiple projects and final exams, we made an oversight that highlighted a critical lesson. Specifically, when testing entropy, we

mistakenly used hashed values as inputs for the test, which invalidated the results. While the outcomes appeared ideal, the data was effectively meaningless.

This mistake, however, became a valuable learning moment. It brought us face-to-face with the mathematical essence of randomness and provided insights into optimizing TRNG implementation.

Additionally, this course allowed me to bridge theoretical knowledge from cryptography and network security into practical hardware implementation. Moving forward, I hope to have the opportunity to design circuits directly tied to these concepts, deepening the connection between theoretical frameworks and real-world applications.

Finally, we would like to extend our heartfelt gratitude to the professor and teaching assistants for providing us with the opportunity to engage closely with industry practices and learn new knowledge. Your guidance has been invaluable throughout this course.

For reference, the font used in our presentation PPT is called **MStiffHeiHK**.

Thank you once again!

［ 吳承翰 ］

In this project, we explored various methods to enhance the randomness of our data. Initially, we considered using a hash function to achieve this goal. However, we overlooked the fundamental definition of an entropy source that our instructor had emphasized in class.

Thanks to our instructor's valuable feedback during the latter part of our presentation, we recognized the flaws in our approach. Specifically, we understood that using a hash function without adhering to the principles of entropy compression could lead to pseudo-random number generation rather than a true entropy source.

With this realization, we revisited and refined our algorithm. The post-presentation

adjustments ensured that our method aligned with the standards for entropy sources, focusing solely on the raw data from the IMU sensor. This iterative process not only improved our methodology but also deepened our understanding of randomness generation and its underlying principles.

We sincerely appreciate teachers's guidance, which was instrumental in steering our project toward a more rigorous and standards-compliant direction.

Thank professor for your dedication throughout this semester. I have learned a great deal from your lectures, especially on the topic of PUF (Physical Unclonable Function). My current lab is conducting research related to PUF, and it occasionally comes up in discussions. However, I had never gained a deep understanding of it until taking this course. Your teaching has provided me with valuable insights into this field, and I am truly grateful for your guidance and instruction.

［ 連正文 ］

This project changed my understanding of hardware security and cryptography. Importantly, is how our initial design's flaws weren't obvious until the presentation, even with good test results. It taught me that security isn't about complex solutions, but rather about understanding and correctly applying basic principles.

The experience showed me that good test results don't automatically mean good design. Learning about entropy, proper use of crypto tools, and the importance of design review was fun. These insights will hopefully help me build better, safer systems in the future.