International Journal of Applied Technology & Leadership
ISSN 2720-5215
Volume 3, Issue 1, January 2024
ijatl@org

Measuring System-on-a-Chip Data Leaks over Radio Transmissions of Small Satellites

Tom Gallagher

Capitol Technical University (USA)

Dr. **Michael E. Johnson** MRAeS

Capitol Technical University (USA)

Abstract

Continuing the topics posed in "Identifying System-on-a-Chip Data Leaks over Radio Transmissions of Small Satellites", this article extends the exploration of data leakage of system-on-a-chip transceivers to quantifying suspected leaks. Specifically, it poses a measurement for the impact of a data leak on the device's cryptographic security. Cryptography is a key security element for satellite communication, so it is important to characterize the impact that data leaks could introduce. In particular, this research contributes to the work posed in "Screaming Channels: When Electromagnetic Side Channels Meet Radio Transceivers." The results show how the proposed measurement is calculated for four system-on-a-chip transceivers.

1. Introduction

As demonstrated by Giovanni Camurati in Screaming Channels: When Electromagnetic Side Channels Meet Radio Transceivers, some mixed-signal system-on-a-chip (SoC) transceivers leak sensitive digital information into analog broadcasts (Camurati et al., 2018, 170-173). Understanding How System-on-a-Chip Data can Leak over Radio Transmissions explained Camurati's work for non-engineers and provided context for the potential impact of the Screaming Channels phenomenon on satellite communications. Identifying System-on-a-Chip Data Leaks over Radio Transmissions of Small Satellites provided a process for detecting unintentional digital noise leakage. However, the presence of a leak doesn't necessarily equate

to an exploitable weakness in a system. There are a variety of factors that can limit the impact of an information leak. Building on the work presented in this series, this article presents a methodology for measuring the impact of a Screaming Channels leak on a cryptographic process used within the leaking SoC. Researchers using this methodology can determine how impactful a leak is for specific configurations of hardware and software.

2. Backgroud

2.1. Revisiting Screaming Channels

The *Screaming Channels* phenomenon is an unintentional modulation of a transmission from a mixed-signal circuit. Mixed-signal circuits have digital components that make calculations and analog components that transmit/receive wireless signals (Das, 2023). The unintended modulation is caused by the digital components' power consumption affecting the voltage of the shared source. Whenever digital components change state (e.g., unpowered to powered), the components cause a small drop or spike in voltage potentially impactful to other components sharing the power source. As analog components are highly susceptible to voltage changes, these variations cause measurable impacts on wireless broadcasts made by these components (Marshall, 2022, 109). In particular, the amplitude of the broadcast fluctuates with the voltage changes. Understanding this interaction and the cybersecurity implications is not the focus of this article but can be found in Part 1 of this series. Instead, this article will explain how a researcher can measure *Screaming Channels* leaks in transceivers used by small satellites.

As demonstrated in *Identifying System-on-a-Chip Data Leaks over Radio Transmissions of Small Satellites*, it is possible to detect leaks that are significantly impacting an analog transmission by alternating the processor between short periods of "sleep" followed by activity. Plotting emitted signals on a time-frequency diagram shows a solid line if the signal is unaffected by the leak and a recognizable pattern, such as a dashed line, if the digital changes are significantly impactful. Figure 1 shows the impact of alternating the processor activity on two devices. The Wio-E5 signal (left) shows no apparent change despite the change in processing, but the nRF52832 (right) shows a visible pattern created by the processor's cycle.

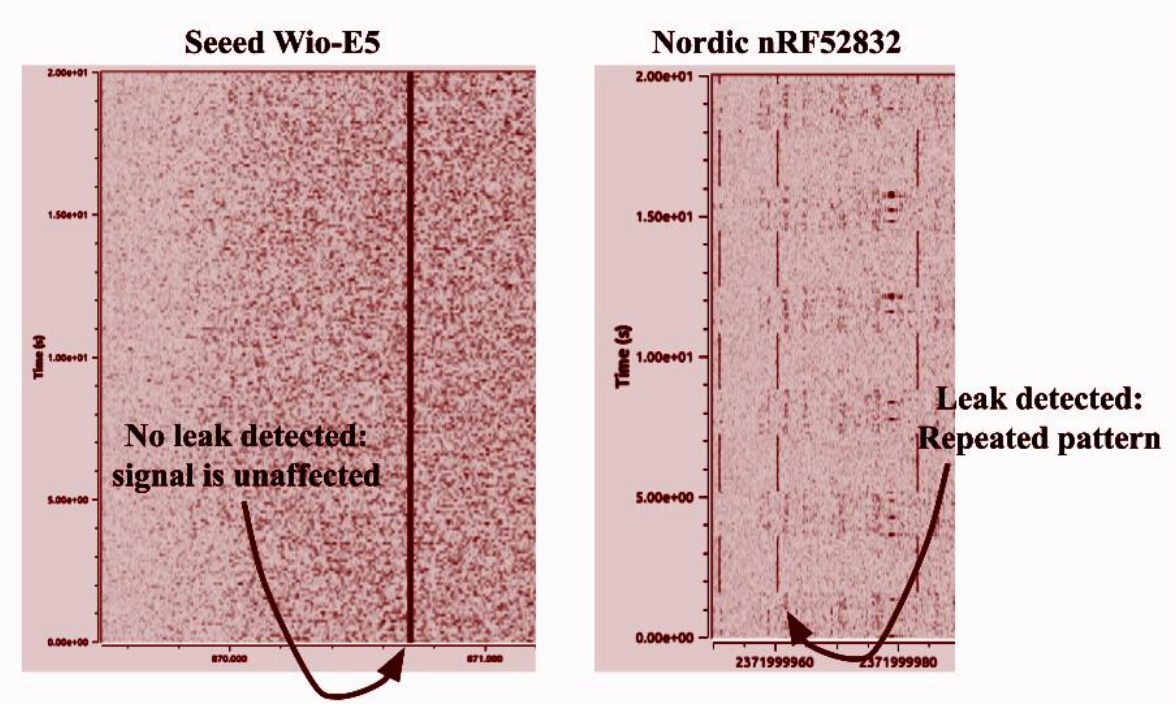

Figure 1: Comparison of time-frequency diagrams from two different transceivers while alternating the processor between periods of sleep and activity. *Note: contrast adjusted for better visibility*

2.2. Cryptography

Cryptography is a fundamental aspect of cybersecurity. When two electronic systems communicate, it could be possible for an attacker to intercept the traffic. Sensitive data across an unencrypted channel is at risk from such interception. This is particularly concerning for satellite communications whose signals can cover more than a third of the Earth's surface (European Commission, 2017). There are a variety of different encryption algorithms having different strengths, weaknesses, and performance characteristics. Generally, the encryption process uses the value of a digital key to alter data such that it is unintelligible until decrypted with the corresponding secret value (see Figure 2). This is a useful paradigm for electronic communication as encrypted data, a.k.a. ciphertext, can be broadcast widely as long as the key is kept secret.

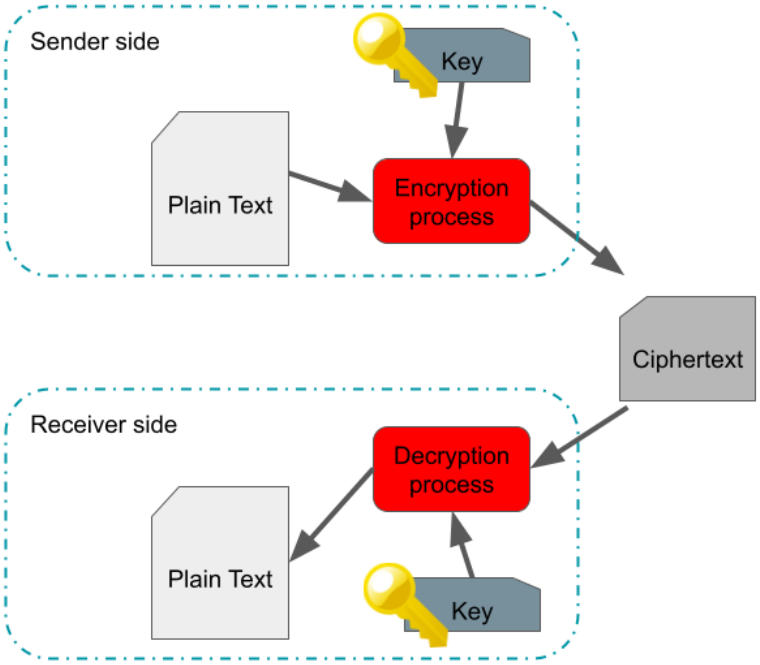

Figure 2: Cryptography in practice

As an easy way to understand encryption, consider this simple example using an "XOR" encryption algorithm. With XOR encryption, each bit of the data is compared to the corresponding bit of the key. If the bits are different, the resultant ciphertext bit is "1", otherwise it is "0".

```
1110011 1101111 1101101 1100101 1110100 1100101 1111000 1110100  Plain Text: "sometext"
1101101 1111001 1110011 1100101 1100011 1110010 1100101 1110100  Encryption Key: "mysecret"
0011110 0010110 0011110 0000000 0010111 0010111 0011101 0000000  Ciphertext: "<XðåÎ"
```

In the above example, the phrase "sometext" is combined with the secret key "mysecret" using the XOR algorithm. The resultant encrypted text is the set of characters "<XðåÎ" [1]. In theory, an attacker who captured the ciphertext cannot understand the original message without the secret key. In this way, if two devices share a secret key, they can communicate securely even when a malicious party is eavesdropping. In practice, the XOR algorithm is extremely weak and susceptible to a number of cryptanalysis attacks (Wells, 2021). Fortunately, modern cryptographic algorithms are stronger than the example.

2.3. Cryptanalysis

Cryptanalysis is the study and practice of decrypting ciphertext without the decryption key, or more simply, breaking encryption to access the plain text. The simplest form of cryptanalysis

[1] Some bytes are non-printable, so the displayed ciphertext is shorter than the plain text

is guessing every possible key value, also known as brute forcing. Consider an unrealistic example where a key is only 4 bits long. In this case, there are only 16 possible values for the key, so it is reasonable to test every possible key value to find the correct one. A modern key size could be 256 bits or more. A key size of 256 bits means that there are 2^{256} possible keys. It is difficult to conceptualize a number as large as 2^{256}, but it is on par with the number of atoms in the visible universe (Read, 2022, 30). For this key size, testing every possible key is not realistic as it would theoretically take 3.3e56 years[2] on a modern supercomputer (Ryan, 2021, 39).

Because brute forcing is unrealistic against large key sizes, cryptanalysis studies algorithms and protocols for subtle weaknesses. For instance, the Triple Data Encryption Standard (3DES) algorithm uses a 168-bit key, but due to a flaw in the algorithm, cryptanalytic attacks can reduce the effective key size to only 112 bits (Schwenk, 2022, 16). Once a cryptanalytic attack is identified against a cryptographic algorithm, it is generally avoided by developers. However, it is challenging to replace cryptography deployed in systems, so many systems still use legacy or even broken cryptography. The National Institute of Standards and Technology (NIST) publishes a number of documents with current recommendations for encryption and digital signatures (NIST, 2023).

Regardless of the key size and security of the algorithm, ciphertext is only as secure as its secret key. If an attacker is able to compromise the encryption key, then decrypting the ciphertext is trivial. While keys are often stored securely within the device, at some point, they are loaded into the processor and used. While they are being used by the processor, keys may be vulnerable to being stolen or leaked, such as through side-channel attacks (Wong, 2021, 291).

2.4. Side-Channel Attacks

Side-channel attacks provide a different attack vector against cryptography. Rather than the cryptanalytic approach of cracking the algorithm without the key, side-channel attacks attempt to glean the value of the key (or some other sensitive data) from unintentional data leaks. Once the key is compromised, any intercepted ciphertext is at risk of decryption. Side-channel attack techniques vary widely from detecting subtle processor emissions to timing millisecond differences between packet transmissions (Kolokotronis & Shiaeles, 2021, 3-41). Generally, side-channel attacks rely on observing some second or third order effect caused by the presence of sensitive data that is otherwise inaccessible to the attacker.

Electromagnetic emissions caused by cryptographic processing are historically vulnerable to side-channel attacks (Schwenk, 2022, 36). While traditional cryptographic side-channels are direct emissions from the cryptographic processor, *Screaming Channels* leaks appear in a signal transmitted by analog components within the device. For this reason, *Screaming Channels* leaks represent a new attack vector for researchers to consider.

[2] 330,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000 years

While side-channel attacks represent a significant cybersecurity concern, the risk is not equivalent for all leaks. Subtle leaks may only expose a single bit of the cryptographic key and be barely detectable above ambient noise. At the opposite end of the spectrum, significant leaks could make every bit of the key detectable over a single encryption process. Depending on the subtlety of the leak, different analysis techniques may have different levels of success. Simple Power Analysis (SPA) is the simplest side-channel attack and measures the power used during a single transaction. If a significant leak is present, the data or key can be inferred directly from the power trace, such as in Figure 3 (Ouladj & Guilley, 2021, 21-22). Because SPA is highly impacted by noise or outside influence, it is effective primarily in tightly constrained environments (Randolph & Diehl, 2020, 3-5). More subtle leaks cause miniscule changes in the signal and may be tied to an internal state of the processor rather than direct key values. Differential Power Analysis and Correlation Power Analysis are analysis techniques that aggregate signal traces and use statistics to infer portions of the key.

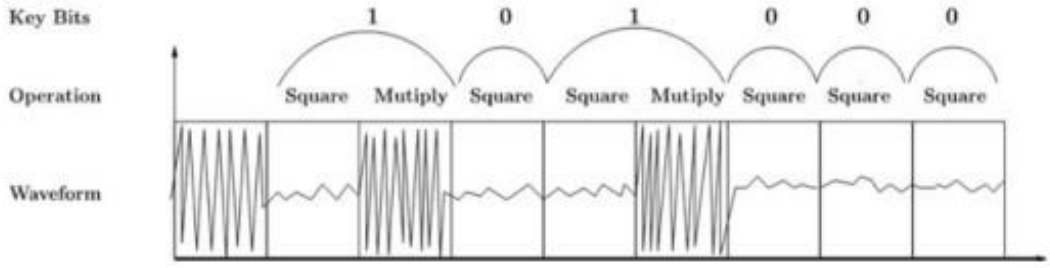

Figure 3: SPA of the Rivest-Shamir-Adleman (RSA) algorithm (Ouladj & Guilley, 2021, 22)

2.5. Cryptographic Characteristics

Cryptographic implementations vary widely in power utilization, speed, and encryption strength. Operational requirements for the device may impose limitations on the encryption used. Satellite hardware in particular tends to minimize power consumption to reduce battery and solar array mass (Vaughan, 2023). Different cryptographic implementations can have an impact on *Screaming Channels* leaks from the device, as described in the sections below.

2.6. Hardware vs Software Implementation

Some SoC transceivers, such as the Texas Instruments CC1310, offer a hardware encryption module separate from the main processor (Texas Instruments, 2020). This gives software developers the choice of implementing encryption within their application using the main processor or offloading encryption to the dedicated module. *Screaming Channels* leaks are due to the interaction of hardware components within the same SoC, so the physical location on the chip where the encryption takes place is potentially impactful. Cryptography running on the processor may cause a noticeable leak while a dedicated module may have negligible impact, or vice versa. From a generic perspective, neither implementation is more or less secure, but

for a specific device, analysis could produce dramatically different results for each implementation. Researchers should be aware of this distinction and use it to contextualize their findings. Depending on the purpose of the research, it may be desirable to focus on only one implementation or analyze each implementation.

2.7. Key Size

Key size has a direct impact on a side-channel leak. Often in side-channel analysis, only part of the key can be recovered through the leak (de Micheli & Heninger, 2020, 2-4). In these cases, if the un-leaked portion of the key is small enough, brute forcing can ascertain the remaining bits. It is intuitive why key size is thus impactful to side-channel attacks. Hypothetically, if a leak exposes half a 128-bit key, an attacker would need to brute force the remaining 18,446,744,073,709,551,616 (2^{64}) possible key values, potentially accomplishable in 776 days on a modern CPU or as little as 22 days on a purpose-built system (Dinaburg, 2019). While still a significant hurdle, this is trivial compared to the task of brute forcing half a 256-bit key, which would theoretically take billions of years using billions of supercomputers (Hoofnagle & Garfinkel, 2022, 212). Researchers should be aware of the impact of key size on this type of analysis. SoC manufacturers often provide libraries or hardware modules that support specific key sizes, but application developers can choose to implement other algorithms with different key sizes. Because key size has a significant operational impact on throughput and power consumption, developers may be restricted by practical limitations, especially in constrained environments like small satellites.

2.8. Side-channel Analysis Resistance

Some cryptographic algorithm implementations have built-in resistance to side-channel analysis. Masking and hiding are well defined practices in reducing the risk from side-channels (Lee & Han, 2020, 1-2). Masking is the process of overlaying random values with sensitive values during intermediate steps of the encryption process and then removing the random overlays. Assuming that the randomized masks are secret, then the intermediate values are protected even if exposed through a side-channel. Hiding is the process of adding random operations into the encryption process. The additional operations introduce challenges for side-channel attacks because the signal traces of a cryptographic process would not directly align with each other. In practice, masking and hiding make side-channel attacks more challenging but not impossible. There have been published attack strategies against masked cryptographic implementations, even recent quantum-resistant algorithms (Backlund et al., 2023, 8-10).

Masking and hiding are provided by some, but not all, cryptographic implementations. Application developers often leverage manufacturer encryption modules or libraries to reduce development costs. Of the four devices analyzed across this series of articles, none included cryptographic implementations with masking or hiding. When manufacturers do not provide resistant implementations, developers can use resistant third-party libraries. However, as with

key size, side-channel resistance introduces additional power consumption and processing time, which may be untenable for small satellites.

2.9. Evaluating Side-Channels

A signal unintentionally emitted from a device does not necessarily constitute a data leak. A data leak implies that the signal carries internal information that can be obtained by an attacker. To show that a signal is actually a leak, it is important to show a relationship between the signal and some private data on the device. The relationship may not be obvious or direct. Instead, the leak could be the result of subtle changes to the signal from which private data can be inferred. While this makes it challenging to verify side-channel leaks, the field of statistics provides valuable tools to show a correlation between the signal and the private data.

One such side-channel evaluation tool, Welch's t-test, assesses whether a binary attribute has a meaningful impact on some characteristic of the population (Bertoni & Regazzoni, 2021, 107). For example, the t-test could help to determine if vegetarians have different heights than non-vegetarians. Having sampled the population for this data, a researcher could put the height into two groups: vegetarians and non-vegetarians. Based on this binary attribute, Welch's t-test could determine the likelihood that there is a statistically significant difference in the average height of vegetarians and non-vegetarians. Figure 4 illustrates this example and how the results of the t-test would be interpreted.

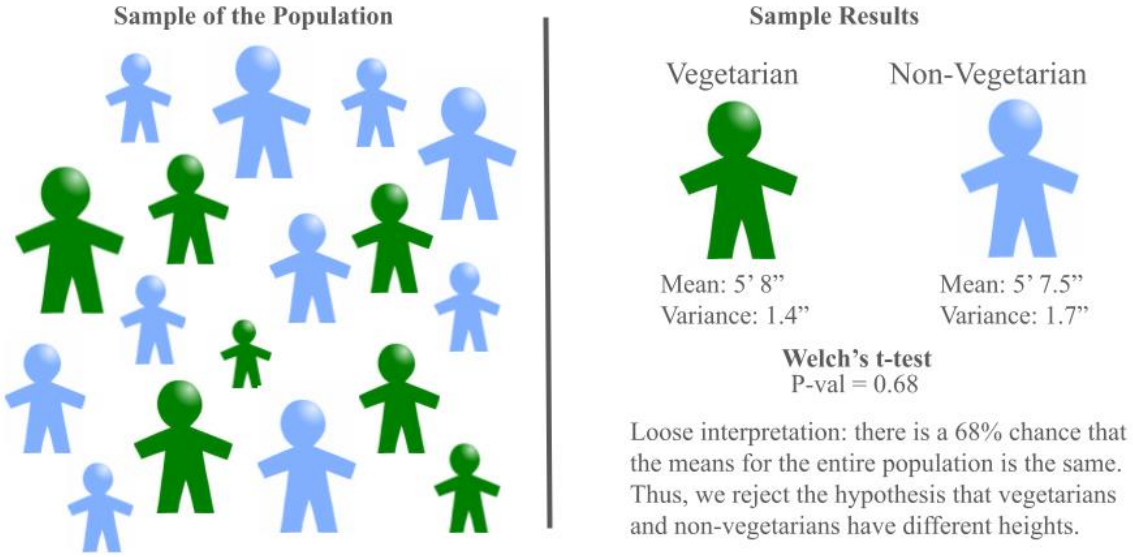

Figure 4: An illustration explaining Welch's t-test to determine whether vegetarians and non-vegetarians have the same average height using fake data

In side-channel analysis, the t-test can be useful in determining whether or not private data is being leaked. The Cryptographic Engineering Research Group (CERG) at George Mason University released the Flexible Open-source workBench fOr Side-channel analysis (FOBOS), which includes a t-test module. FOBOS leverages the t-test to determine whether or not the

value of the private encryption key has a detectable impact on a non-private power trace (CERG, 2019). To perform this test, the target device performs a number of encryptions using a single encryption key and encryptions using a random encryption key. The power traces from each encryption are categorized as using the fixed key or a random key, which is the binary attribute for the t-test. The t-test determines if there is a statistical difference between the two groups at any given time slice of the power traces. If so, then the key value is impacting the signal, or stated differently, the power signal provides information about the private key, which is the basis of a data leak.

Similar to the fixed-vs-random key t-test, it is possible to more closely examine the impact of specific key/plaintext bits. These more detailed bitwise examinations are referred to as leakage models. Leakage models show how the cryptographic key and plaintext are related to the leaked signal. There are a wide variety of leakage models including least-significant-bit (LSB), least-significant-2-bits (LS2B), most-significant-bit (MSB) and Hamming weight (HW) (Zhou et al., 2022, 222-223). Shown in Table 1, each model describes a categorization based on a single byte of the key and/or plaintext. LSB and MSB look at a single bit of a target byte providing two groups (where the bit is 0 and where it is 1). LS2B looks at two bits of the target byte creating four groups (00, 01, 10, and 11). Hamming weight counts the number of 1's in a target byte of the key resulting in 9 groups with values 0 through 8. More complex leakage models can be categorized based on a relationship between the key and plaintext, such as Hamming Distance, which counts the bit flips between the key and plaintext.

Leakage Model	Example: 010110010		Example: 11100101	
	Salient bits	Group	Salient bits	Group
LSB [Possible groups: 0,1]	01011001**0**	0	1110010**1**	1
MSB [Possible groups: 0,1]	**0**10110010	0	**1**1100101	1
LS2B [Possible groups: 00,01,10,11]	0101100**10**	10	111001**01**	01
HW [Possible groups: 0,1,2,3,4,5,6,7,8,9]	0**1**0**11**00**1**0	4	**111**00**1**0**1**	5

Table 1: Example of LSB, MSB, LSB2, and Hamming Weight

Similar to how Welch's t-test provides a metric regarding whether two groups are related, leakage models determine if the grouping has a statistically significant impact for the target byte. A useful metric for measuring the impact of the groupings is the signal-to-noise ratio (S2N). A signal-to-noise ratio compares the useful information (the signal) to undesirable information (the noise). For the purposes of side-channel analysis, useful information is any observable difference in the signal amplitude between the leakage model groupings (Buhan & Mangard, 2021). The not useful information is any deviation that is not related to the trace's leakage group. Put simply, a high S2N value indicates that the categorization provides

meaningful information, which means the signal is a leak, and a low S2N value indicates that the categorization is not meaningful.

3. Measuring Screaming Channels Leaks

3.1. Assumptions

Broadcasting of radio transmissions must adhere to legal restrictions for the locality in which the transmission is made. Additionally, researchers must be authorized to perform testing for a target device. These concerns are covered extensively in *Identifying System-on-a-Chip Data Leaks over Radio Transmissions of Small Satellites*. The process described here assumes that researchers perform analysis legally and ethically.

3.2. Software Development

Background on developing firmware for small satellite SoC transceivers was provided in *Identifying System-on-a-Chip Data Leaks over Radio Transmissions of Small Satellites*. This section describes additions to the firmware development proposed in that article. The additions described here are more complex than those described in the prior article and could require additional time and expertise in development.

For direct control of the timing for exercising the process, the firmware should ideally provide a wired communications path. Wireless communications should be avoided as they could impact the signal being analyzed. Universal Asynchronous Receiver/Transmitter (UART) is a communications protocol frequently supported by SoCs. SoC development boards may expose a UART interface over Universal Serial Bus (USB). Regardless of the communications path used, it should provide a low latency method to programmatically trigger functionality on the device, pass data to the device, and receive data from the device.

To produce the data required for this analysis, the application logic for the firmware should listen on the wired communications path for commands from the controlling device. Specifically, the firmware should be able to perform a given number of encryption processes with a specified key/plaintext. The application should also deliver status updates back across the communications path, such as when the encryption process completes. Additionally, if more than one encryption implementation is to be tested, such as for devices that provide both hardware and software implementations, then it is valuable for the application to accept commands that dictate which implementation to use.

Below is pseudocode for the firmware's application logic discussed above[3]:

```
main() {
      CypherText = "";
      initBoard(); //enable general device hardware
```

[3] Source code for the devices analyzed in this research is available through the link in *Contributions*

```
        initRFBoard(); //enables the Radio


        disableSpecialRFFeatures(); //such as Frequency hopping
        SetFrequency(900MHz);
        SetPower(11); //Power levels are vendor specific
        StartContinuousRadioTransmit(); //transmit a dummy signal
        while(1) { //loop this logic until device is powered off
              SendOutput("Awaiting Command")
              Command = AwaitExternalInput()
              If (Command = "Encrypt") {
                    Repetitions = AwaitExternalInput()
                    Algorithm = AwaitExternalInput() //If more than one
                   Key = AwaitExternalInput()
                   Plaintext = AwaitExternalInput()
                   SendOutput("Starting Encryption")
                   For(Repetitions) {
                         CypherText    =    Encrypt(Key,    Plaintext,
Algorithm);
                   }
                   SendOutput("Encryption Complete")
              }
        }
}
```

Unlike the firmware developed previously in this series, the firmware described here now depends on an external controller. This external component is a separate application script that coordinates capturing traces during encryption periods. The logic for this script is straightforward, but robust testing should be performed as complexities arise when passing data between devices. A key aspect of the controlling script is correctly configuring the frequency and gain of the software-defined radio receiver. The GNU Radio Companion (GRC) user interface provided in the prior article can be used to identify acceptable values for these settings. The importance of selecting appropriate gains should not be overlooked. Gain amplifies the received signal at various stages of its processing. Too little gain and the received values will be insignificant; too much gain and background noise overpowers the signal. Tailoring the receiver to the signal under analysis is an iterative process. While this iteration is part of the collection/analysis process, developing a controlling script that can easily modify these values is beneficial.

Below is pseudocode for the external controller script discussed above[4]:

```
Traces_to_capture = X
main() {
      EstablishCommsWithTargetDevice()
      EstablishConnectionWithReceiver()
      ConfigureReceiver(TargetFrequency, Gain)
      While(GetTargetDeviceResponse() != "Awaiting Command") {
            Wait
      }
      For(Traces_to_capture) {
            Key = Random()
            Plaintext = Random()
            Receiver_StartTrace()
            SendCommandToTargetDevice("Encrypt")
            SendCommandToTargetDevice(1) //Repetitions
            SendCommandToTargetDevice(Key)
            SendCommandToTargetDevice(Plaintext)
            While(GetTargetDeviceResponse() != "Encryption Complete")
      {
                  Wait
            }
            Trace = Receiver_CompleteTrace()
            Save(Trace, Key, Plaintext)
      }
}
```

The guidance in this section recommends one approach to develop firmware capable of supporting this analysis. Alternate approaches could be necessary depending on the target device. For instance, if a wired communications path is not feasible, then it may be necessary to use different means to trigger the encryption process. Different collection techniques will have varying impacts on the resultant traces, such as trace alignment, and could require additional effort during analysis.

[4] Source code for the devices analyzed in this research is available through the link in *Contributions*

3.3. Collecting the Data

With the external script and firmware configured, the target device located inside an RF enclosure is attached to the external command laptop. A HackRF One software-defined radio is connected to the laptop with its antenna connected inside the enclosure, see Figure 5. The laptop executes the script and records a set number of traces. Initially, while establishing a test environment, hundreds of traces are sufficient to show the configuration is functional. For performing the analysis, the greater the number of traces, the more accurate the results. Sizes of traces will vary depending on the device setup and collection methodology, so researchers will have to account for storage and processing time based on their resources.

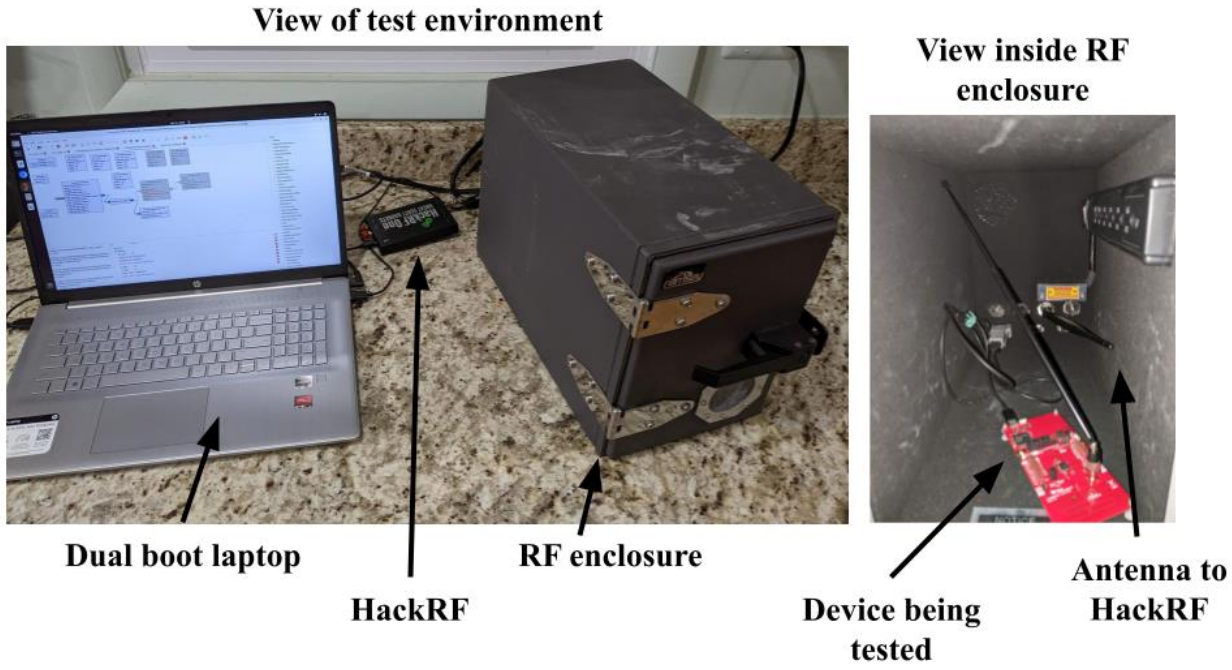

Figure 5: The environment for collecting traces

3.4. Analyzing the Data

This section is intended to provide an outline for the process of analyzing traces to measure the impact of a detected *Screaming Channels* leak. It would be easy to assume that such a process is linear, and that the severity of a leak could be determined through a set number of steps. In reality, the process requires iterative testing of configurations to optimize visibility. It may be challenging to perform this analysis without prior experience. It may be helpful for researchers to analyze a known-vulnerable target prior to assessing an unconfirmed leak. Understanding the output from a known leak will aid in interpreting the results of an unconfirmed leak. For ease of entry into this research, data collected from this study is available through a link in the *Contributions* section of this article. Researchers can gain experience in analyzing the provided data before assessing a new target.

An optional first step in analyzing the data is trimming each trace to focus on the segment containing the cryptographic process. There is a significant lag created by initialization/closeout

of the software-defined radio (SDR) and communication between devices. The lag results in a larger than necessary trace that captures uninteresting data both before and after the cryptographic processing. While it is possible to complete the analysis against the full range of the trace, such processing can create unrealistic hardware requirements or drastically increase the analysis timespan. However, it is difficult to predict where in the trace the cryptographic process occurs. In some cases, the cryptographic process may cause a perceptible change in the trace such as a drop in amplitude allowing researchers to easily hone in on the portion of interest. It is also possible to trigger visible changes in the signal immediately prior to and after the encryption process; altering transmission frequency or broadcast power are two such techniques. Figure 6 shows how spiking the signal strength immediately before and after the encryption process highlights the interesting part of the trace. In the case of the CC1310, the sample trace is reduced from approximately 800,000 samples to around 20,000 samples, resulting in a 97% savings in storage and processing.

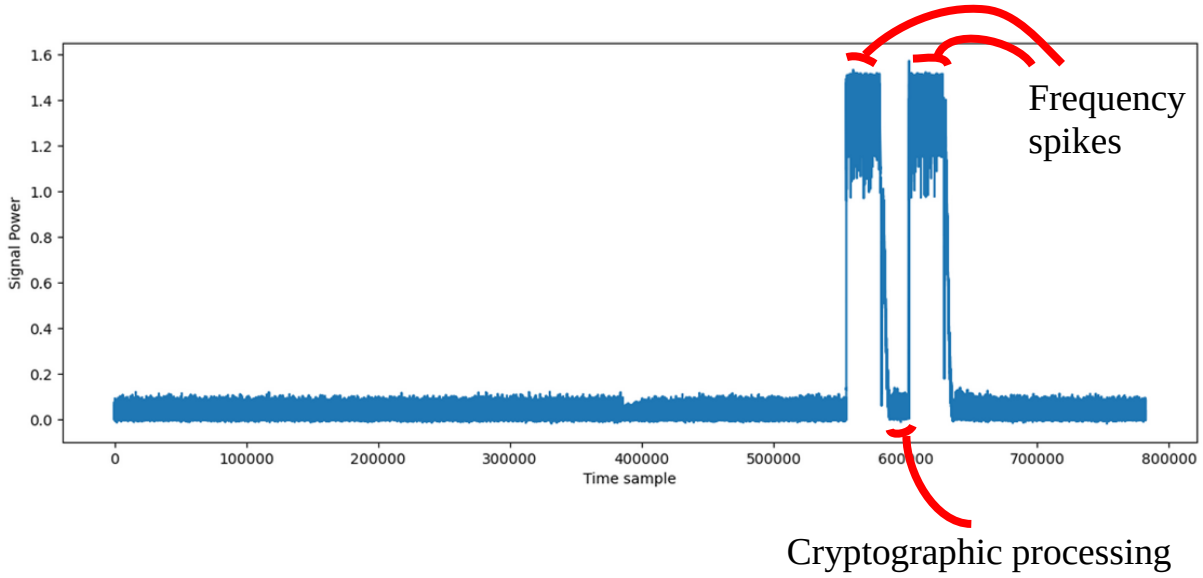

Figure 6: A full traces of the Texas Instruments CC1310 using signal spikes to highlight the start and end of the encryption process

Each trace is a series of snapshots measuring the signal strength at a specific time. The analysis compares the x^{th} sample of each trace and assumes that the device was doing the same thing at each time across all traces. For instance, the 316^{th} sample of each trace could be the middle of the first round of encryption and the 1099^{th} sample may be the start of the last round of encryption. The device performing the same action at any given trace index is a core assumption of the analysis. In an ideal world, because every trace is generated by the same process, they would be completely aligned. In reality, the collection process introduces small variability in each trace. Failure to properly account for this misalignment can distort the results. Signal alignment is a complex topic with a variety of proposed solutions (Pearson et al., 2019, 21). Using an RF enclosure reduces external noise while the full process control provided by the custom firmware provides consistency across traces. Using a script to ensure that traces are properly aligned is recommended. An alignment script is available in the *Contributions* section of this article.

With well-aligned traces, the analysis can continue with evaluation of the leakage models. Leakage models label each trace according to key and plaintext values for that trace. The analysis then determines if that leak model categorization is impactful to the set of traces at each time slice. For example, consider a hypothetical case where each trace is binned according to whether the first bit of plaintext is a 0 or a 1. If analysis showed that the variance between the groups is greater than the ungrouped variance, then there would be a meaningful difference in the signal that correlated to the first bit of plaintext. The correlation is greater if the variance between the group's means is larger or if the noise is smaller. There are a wide variety of possible leakage models that can be used to categorize the traces. Once categorized, the process of calculating the signal-to-noise ratio is the same. Below is pseudocode to calculate the SNR for a leakage model discussed above[5]:

```
GroupMeans = []
TraceGroups = OrganizeByLeakageModel(Traces)
For (group_id, group_traces in TraceGroups){
      GroupMeans[group_id] = CalculateMean(group_traces)
}
MeanVariance = CalculateVariance(GroupMeans)
For (trace_id, trace in Traces) {
      TraceDifferences[trace_id] = trace - GroupMeans[GetGroupID(trace)]
}
DifferenceVariance = CalculateVariance(TraceDifferences)
S2N = MeanVariance / DifferenceVariance
```

Once the SNRs are calculated, it is useful to view them graphically. The Python module matplotlib is commonly used in data science to provide this capability. While the visual of SNR over time for a leakage model is useful, it does not provide an overall measurement for a potential leak. This article proposes an analytic to measure the impact of a leak on a device's cryptography. SNR can be impacted by a number of factors creating biased results. To measure the significance of a leak, it is useful to compare the SNR of the potential leak to the background SNR while the leak is not present. This article proposes that such a measurement can be accomplished by comparing the mean of the background noise to the maximin SNR. This ratio, hereafter the leak ratio, would be computed by dividing the max SNR by the trimmed mean of the SNR values. A trimmed mean is the average of a set of values after removing some number of the greatest and least values from the set. The formula for calculating the leak ratio would thus be:

```
LeakRatio = Max(SNRs) / TrimmedMean(SNRs)
```

[5] Source code is available through the link in *Contributions*

A challenge in calculating the leak ratio is in determining the portion of data to remove for the trimmed mean. A simple way of accomplishing this would be to collect significantly larger traces than needed to ensure that the volume of samples during normal processing dwarfs the samples containing cryptographic processing. An alternative solution would be to include logic during collection to time the encryption routine and ensure that its time slice fits within the trimmed data. For example, if using a 20% trimmed mean, the encryption process would need to last no longer than 20% of the trace's total time.

4. Results

4.1. Nordic Semiconductor nRF52832

During the 2018 *Screaming Channels* research, the Nordic Semiconductor nRF52832 was shown to exhibit a leak across its broadcast signal. Additionally, the framework provided in *Identifying System-on-a-Chip Data Leaks over Radio Transmissions of Small Satellites* similarly showed strong evidence of a leak in the device. As a Bluetooth transceiver, the nRF52832 is not a viable communications device for small satellites, but including this device in the analysis provides a baseline for how an exploitable leak appears in the analysis results.

The other benefit to including the nRF5232 in this analysis is that the *Screaming Channels* researchers shared traces from the original study in a public repository (Camurati, 2021). Because the provided traces use random plaintext and a random key, only the SNR analysis can be performed; a t-test requires a group of "fixed key" traces as well. The traces were analyzed using a Jupyter Notebook that implements the previously described analysis. Figure 7 shows that each analyzed leakage model displays a clear spike in the signal-to-noise ratio at approximately the same time slice. Every byte[6] of the 128-bit key has a unique SNR for each model, meaning that different models could be more or less accurate in predicting specific bytes. Overall, there is a striking difference between the SNR of the non-cryptographic processing and the cryptographic processing, which begins around time slice 850 in the figure.

[6] A byte is an 8-bit segment of data

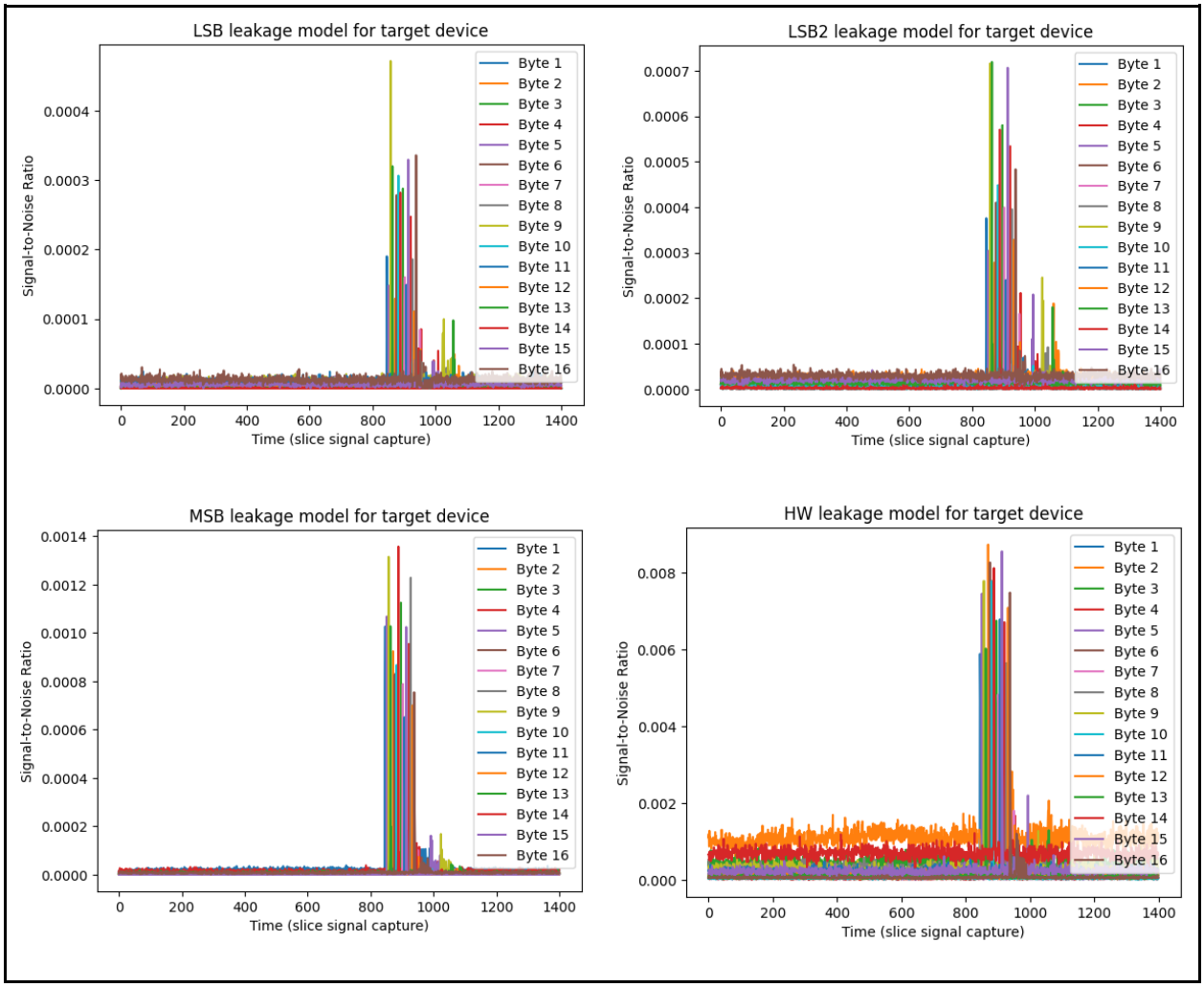

Figure 7: SNRs for the nRF52832 traces using the LSB, LSB2, MSB, and HW leakage models.

Visually, it is clear that the SNR during the cryptographic processing reaches more than 10 times that of other processing. Using the formula proposed above, the leak ratio can be calculated with the results in Table 2. While different bytes of the key are exposed to varying degrees by each of the leakage models, aggregating these metrics shows that the MSB leak model provides an average leak ratio of 135.3. Because the 2018 *Screaming Channels* research proved that the cryptographic key can be compromised across this leak, it stands to reason that devices with similar or larger leak ratios are at higher risk to such compromise. Side-channel analysis and cryptanalysis are complex fields with evolving techniques and challenges, so a high leak ratio alone should not be taken as proof that a leak is exploitable. Instead, the leak ratio provides one metric that can be calculated without extensive background in the field.

	LSB	LSB2	MSB	HW
Byte 1	74.4	4.3	36.8	11.8
Byte 2	55.1	12.9	88.5	4.2
Byte 3	12.9	7.7	201.1	11.2
Byte 4	62.0	8.5	119.9	16.7
Byte 5	44.9	22.5	88.6	18.0
Byte 6	6.9	6.1	38.2	7.4
Byte 7	6.9	6.2	34.7	5.9
Byte 8	43.5	7.7	291.7	4.3
Byte 9	38.9	20.2	207.2	3.7
Byte 10	7.2	6.5	287.7	14.2
Byte 11	18.6	13.3	195.7	12.1
Byte 12	14.0	5.1	17.5	15.4
Byte 13	59.2	20.3	183.3	5.2
Byte 14	5.2	6.3	99.3	9.4
Byte 15	14.4	5.5	104.1	13.0
Byte 16	33.0	9.9	170.2	24.5
Average	**31.1**	**10.2**	**135.3**	**11.1**

Table 2: Leak ratio of LSB, LSB2, MSB, and HW for nRF52832

Because the traces provided by the original researchers did not include fixed-key/fixed-plaintext traces, a Welch's t-test could not be performed against these traces. In order to accommodate a t-test for the nRF52832, additional traces were collected for this article. The samples were collected, filtered, and aligned according to the process described above. After aligning the samples, the samples were able to be processed by the t-test. For this test, two sets of 10,000 traces were collected where one set uses a single encryption key for every trace and the other uses a random key for every trace. The t-test measures the probability that the averages of each group are the same. If the averages are expected to be different, then we have evidence that the key value has an impact on the signal. Figure 8 graphs the probability-value (pval) from the t-test at each time slice. Based on the values in this graph, a large section of the trace, approximately from sample 310,000 to sample 350,000, the pval is generally very low, less than 0.1e-7 (i.e., 0.00000001). Despite appearances, this is not a bar graph; there are multiple values at the bottom and top of the graph resulting in consecutive vertical lines. The values in this segment of the trace are likely to be influenced by the value of the encryption key, so there is strong evidence that a leak is present. It stands to reason that this is the encryption timespan.

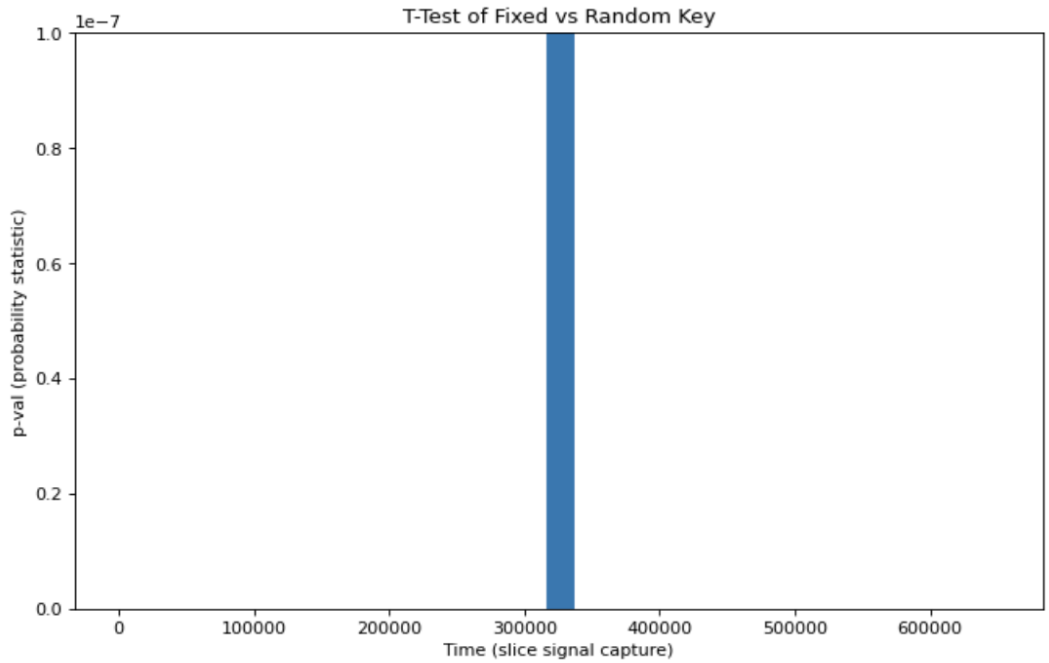

Figure 8: Welch's T-test comparing variable and fixed key traces of the nRF52832. *Note that the y-axis is clipped at 1e-7 and all non-visible data is above that threshold.*

Performing cryptanalysis on the leak is beyond the scope of this article and was already completed during the 2018 study. However, if this was a device with an unverified leak, then the calculated leak ratio would suggest a likelihood that the leak is significant. The t-test results would highlight the trace segment during which the leak is most likely to occur.

4.2. Texas Instruments CC1111

As discussed previously in this series, the CC1111 is nearly identical to the CC1110 used in the Tartan Artibeus small satellite. *Identifying System-on-a-Chip Data Leaks over Radio Transmissions of Small Satellites* identified a potential *Screaming Channels* leak in the CC1111 at 868.5 MHz. The CC1111 was the only device, other than the known vulnerable Bluetooth transceiver, which showed significant indications of a leak according to the proposed detection technique.

Due to development environment licensing issues, the firmware for the CC1111 was not developed with a communications path to the command laptop. To capture the required traces despite this limitation, the firmware was configured to loop the following process at startup: turn radio transmitter off, wait 50 milliseconds, turn radio transmitter on, perform single encryption, and finally wait 1 millisecond. The command laptop initiated a trace on the SDR at the target frequency and then the CC1111 was powered on. Because of the visible impact of enabling and disabling the radio, as seen in Figure 9, it was straightforward to programmatically separate the aggregated traces into individual samples capturing periods where the transmitter was on. A Python Jupyter Notebook script was used to separate the traces and is provided in the repository linked in the *Contributions* section. The most significant issue with collecting samples in this manner is the potential for trace misalignment during the separation process.

Because the amplitude spikes in the trace happened at precise intervals, there is confidence that separation did not cause significant misalignment. Additionally, the signal alignment routine of the analysis can correct minor alignment issues.

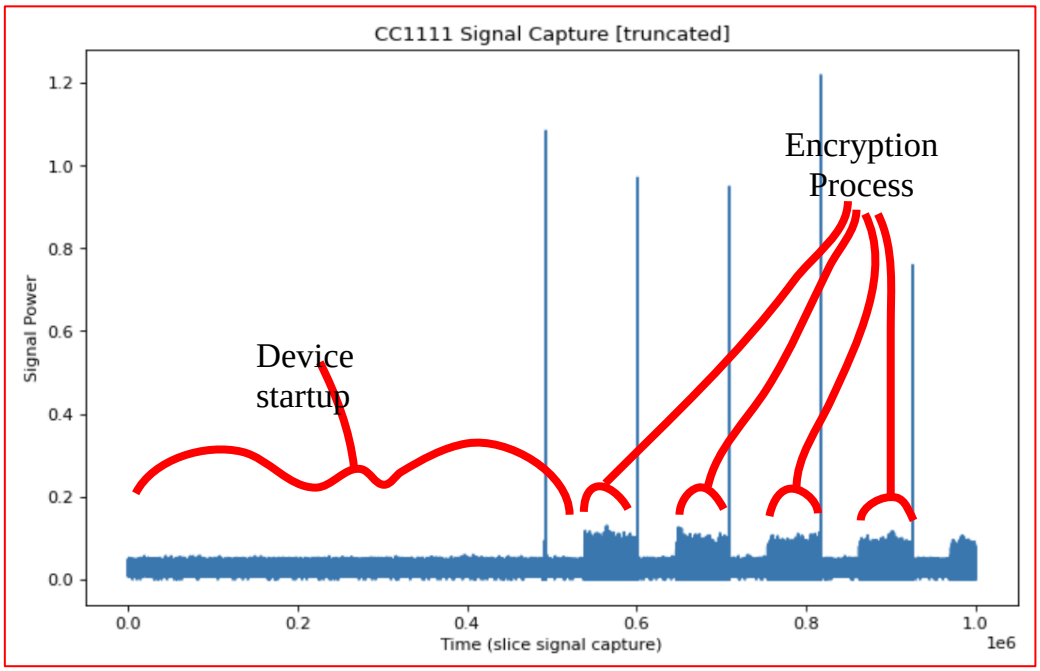

Figure 9: Four encryption cycles of the CC1111 showing distinction between each cycle

Welch's t-test of variable vs fixed keys showed interesting results for the CC1111. As can be seen in Figure 10, there are many areas in the trace where the t-test pval drops below 0.1e-7. A low p-value is evidence that the values were likely to be generated by different systems. Ideally, low pvals would be expected only when the value of the key was impactful to the resultant signal. In the case of the CC1111 capture, the encryption process occurs at the end of the collection, so there should be no correlation with the key across the entire trace. Instead, it is likely that the process of creating/collecting the traces introduced bias across the trace. However, it is noteworthy that the low pvals are more dense toward the end of the trace, around slice 35000, when the cryptographic process is expected to occur. If the CC1111 traces could be collected using the recommended process, it is likely that the low pvals outside of the encryption window would mostly or entirely disappear.

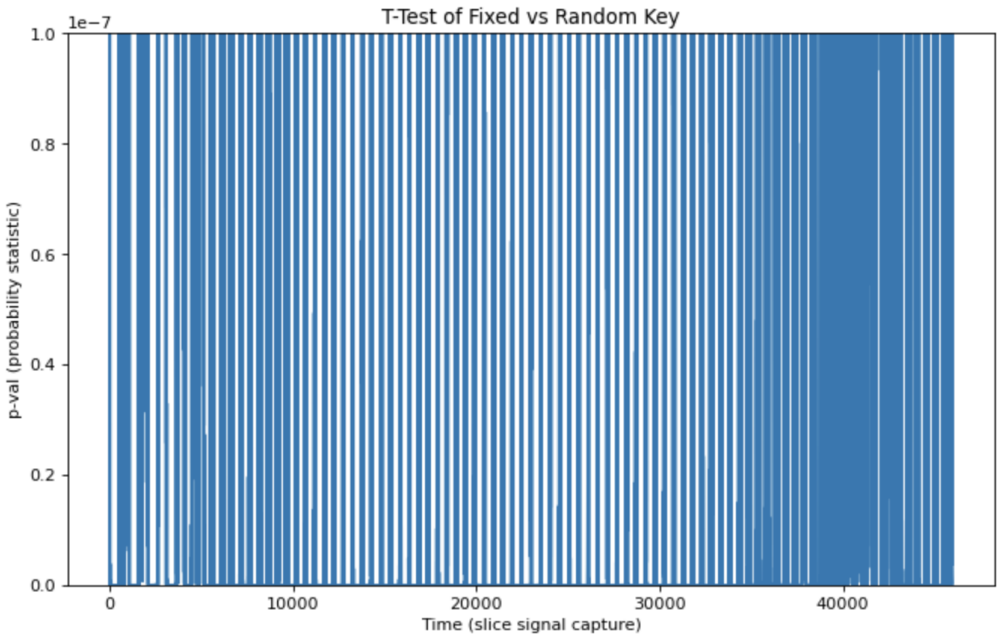

Figure 10: Welch's T-test comparing variable and fixed key traces of the CC1111

Unlike the nRF52832, the CC1111 did not have leakage models that showed significant SNR spikes during the encryption process. Figure 11 graphs the SNRs during the entirety of the trace and shows no clear segment where the SNR is higher than the rest of the trace. As expected, these SNRs yielded low leak ratios as well. The average leak ratios for the LSB, LSB2, MSB, and HW models were 33.4, 10.3, 33.7, and 12.7 respectively. While the t-test shows a potential area of interest, the leak ratio for this device is relatively low. Performing cryptanalysis to verify the exploitability of such a leak is outside the scope of this research. Based on the measurements proposed here, the leak is significantly weaker than the nRF52832 and would likely require additional traces or alternative techniques to successfully compromise the key if possible.

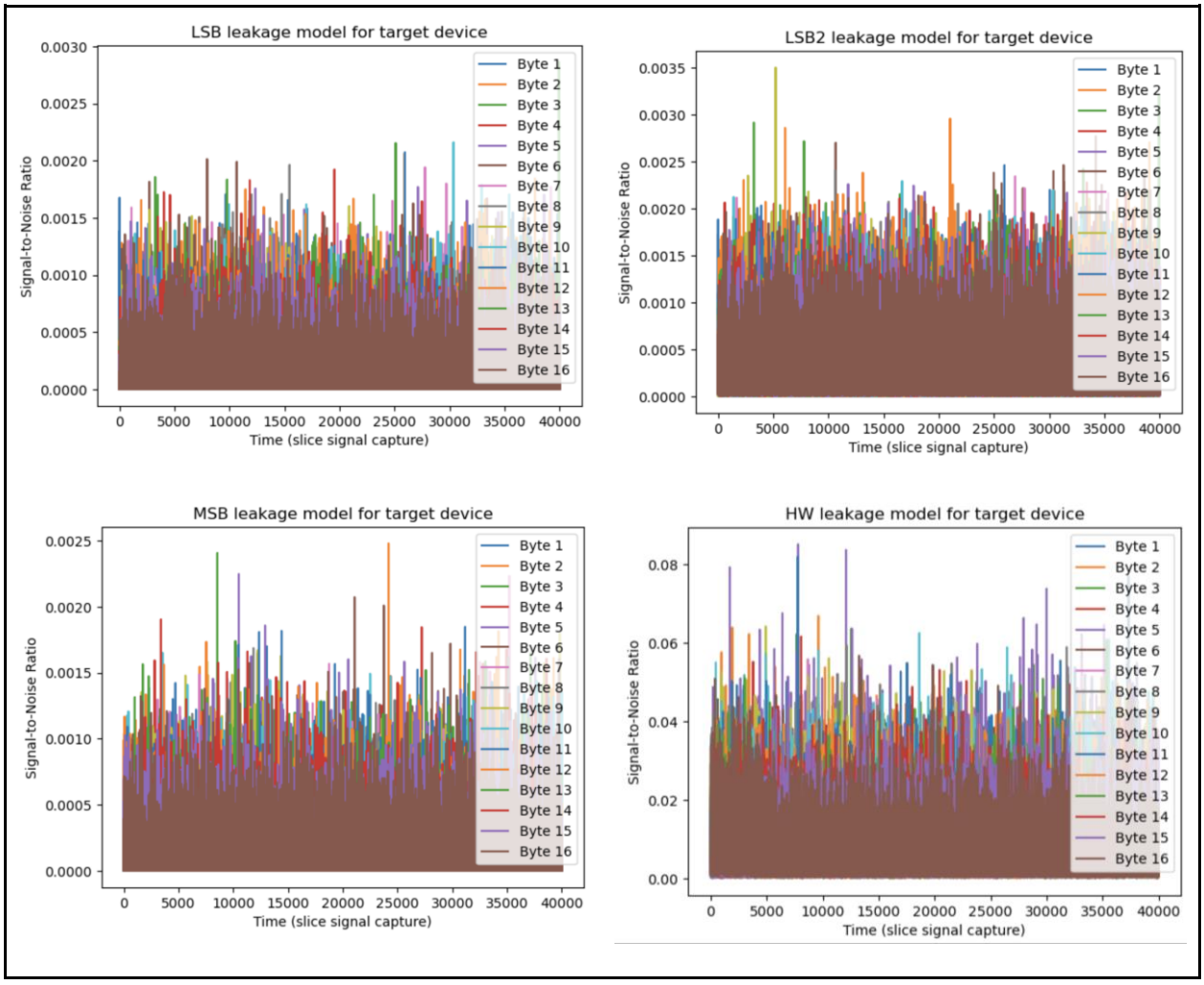

Figure 11: Computed SNR for the CC1111 using the LSB, LSB2, MSB, and HW leakage models.

4.3. Texas Instruments CC1310

As discussed previously in this series, the CC1310 is used in the Monarch chip satellite. *Identifying System-on-a-Chip Data Leaks over Radio Transmissions of Small Satellites* identified anomalous signal artifacts for the CC1310 at 847.9 MHz. These anomalies did not adhere to the framework's prediction for a *Screaming Channels* leak, but the signal will be evaluated here to determine if a leak is present.

To support this research, encryption traces for the CC1310 were captured and aligned using the recommended process. Welch's t-test computed the likelihood that the value of the key was impactful of the resultant signal. As seen in Figure 12, several pvals from the t-test were low, below 0.1e-7. These segments of low pvals could indicate when encryption is taking place and impacting the resultant signal.

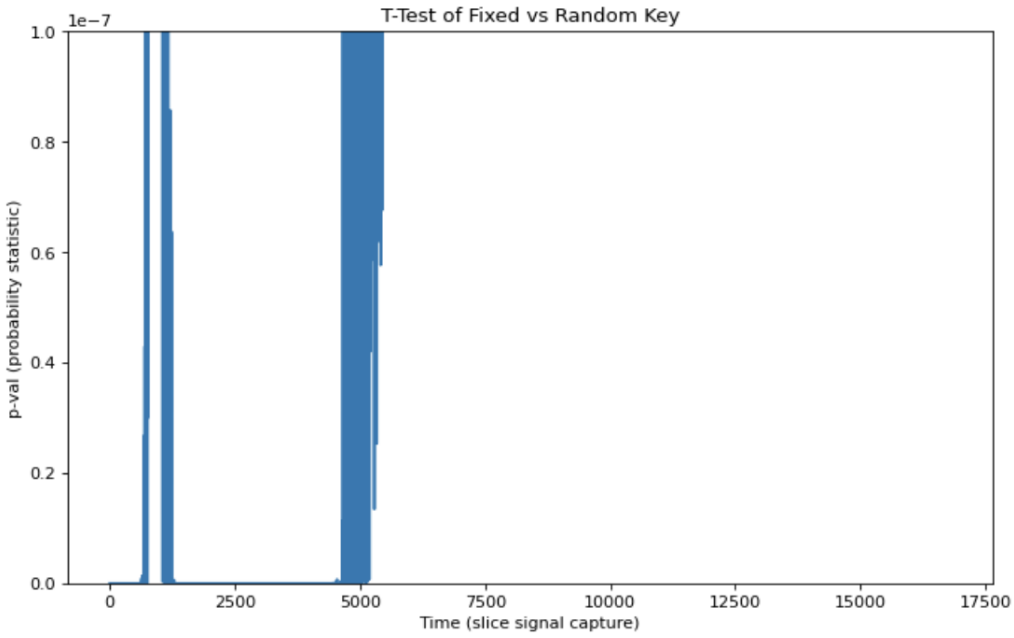

Figure 12: Welch's T-test comparing variable and fixed key traces of the CC1310

Similar to the CC1111, the CC1310 did not show significant SNR spikes during encryption. Figure 13 graphs the SNRs during the entirety of the trace and shows no clear segment where the SNR is higher than the rest of the trace. The average leak ratios for the LSB, LSB2, MSB, and HW models were 30.2, 9.6, 28.3, and 11.2 respectively. Like the CC1111, the leak ratio indicates that the device is likely more challenging to successfully extract a key than the nRF52832. Based on the t-test, there is a clear segment for cryptanalysis researchers to focus efforts.

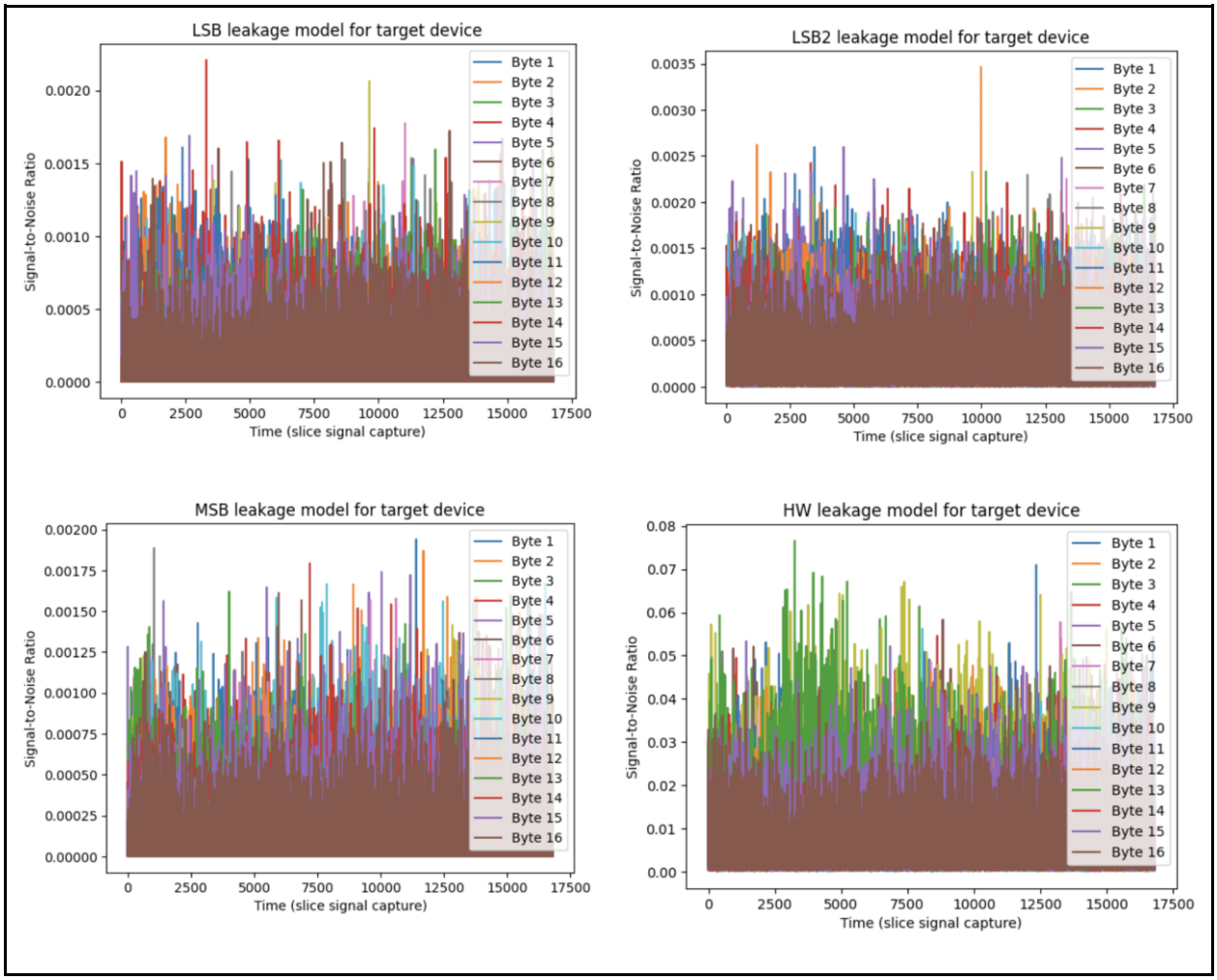

Figure 13: Computed SNR for the CC1310 using the LSB, LSB2, MSB, and HW leakage models.

4.4. Seeed LoRa Wio-E5

The Seeed LoRa Wio-E5 provides similar capabilities to the transceiver used by the AmbaSat project. *Identifying System-on-a-Chip Data Leaks over Radio Transmissions of Small Satellites* identified unintentional broadcasts by the device but none that correlated to patterns predicted from *Screaming Channels* leaks. It is unlikely that the analysis performed here will identify a leak, but the device is included for the sake of completeness.

Like the CC1111, the collection process was not implemented using the recommended approach due to issues in establishing a wired communication route between the command laptop and the device. Instead, the device was configured to loop through transmit, encrypt, and sleep patterns. Also similar to the CC1111, this alternate collection method appeared to introduce anomalous artifacts as the t-test shows low pvals across most of the trace, Figure 14.

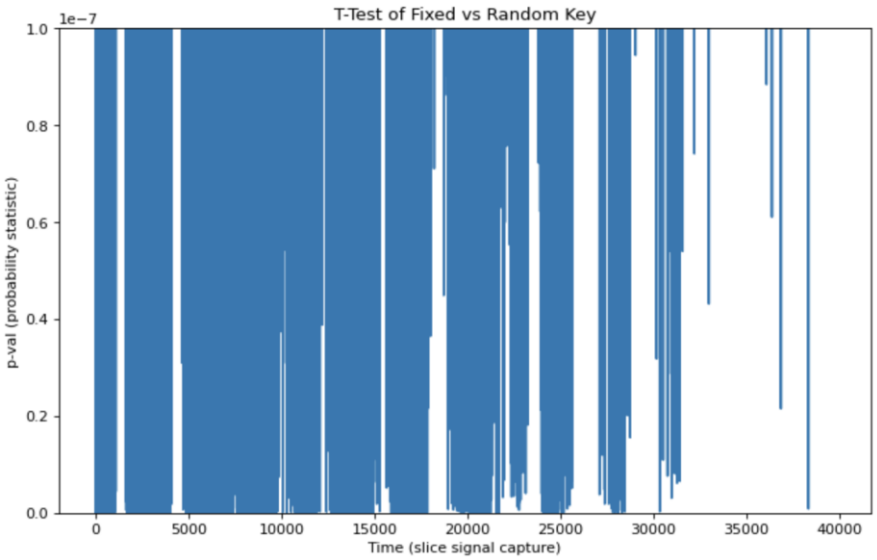

Figure 14: Welch's T-test comparing variable and fixed key traces of the Wio-E5

The Wio-E5's results from the SNR analysis and leak ratio calculation were similar to those of the CC1310. Figure 15 shows the SNRs of the leakage models with no clear indication of the encryption process. Average leak ratios for the LSB, LSB2, MSB, and HW models were 31.2, 9.8, 30.6, 12.4 respectively. Because the previous article showed no indication of a *Screaming Channels* leak for this device, it is possible that these leak ratios could indicate a lower risk of an exploitable leak.

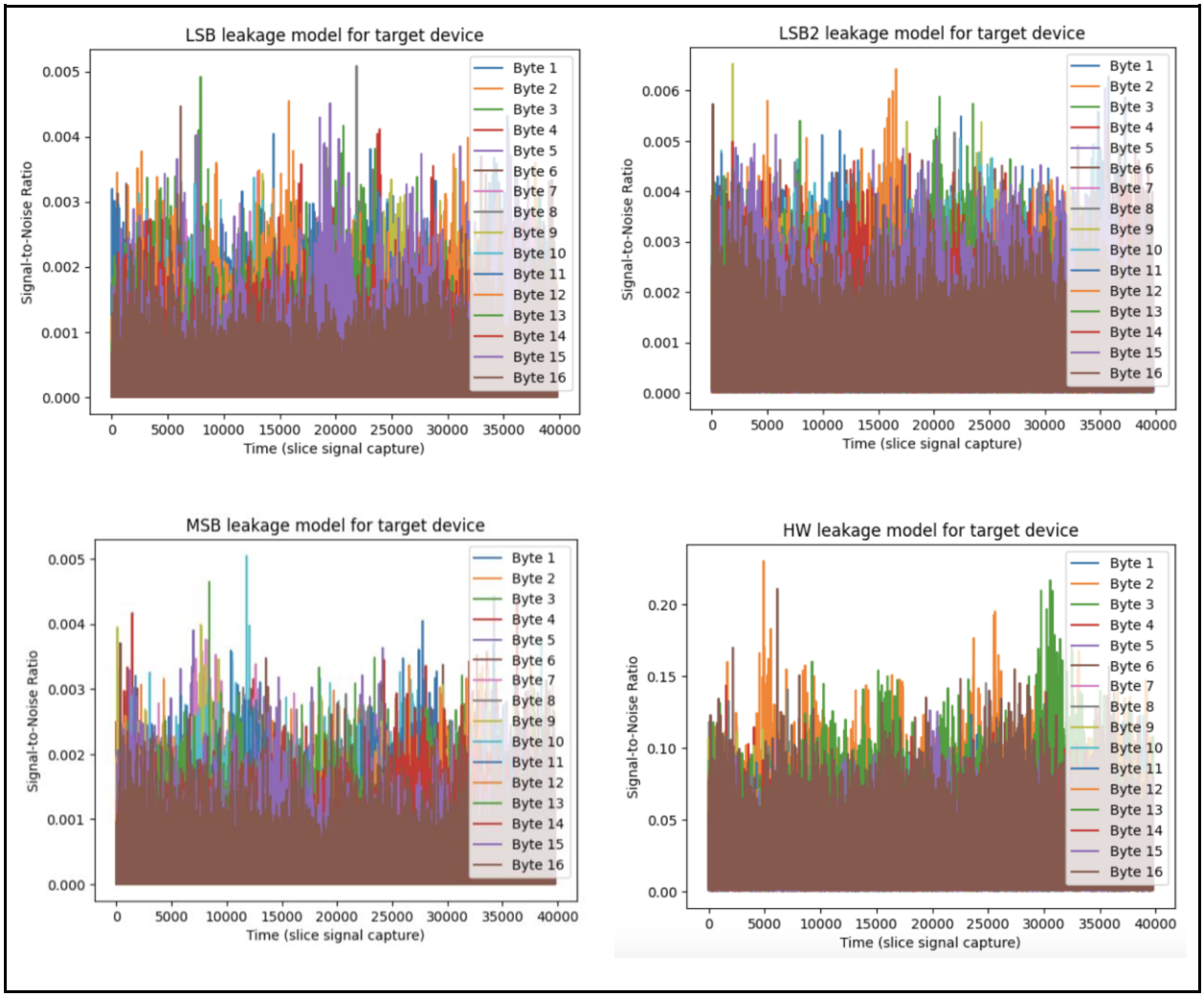

Figure 15: Computed SNR for the Wio-E5 using the LSB, LSB2, MSB, and HW leakage models.

5. Countermeasures

Screaming Channels is a recognized vulnerability. Both hardware manufacturers and software developers share responsibility in ensuring SoC transceivers are resilient against this potential weakness. Attackers do not depend on a single attack vector; likewise, defenders should avoid complacency of relying on a single defense mechanism. Defense in depth is the strategy of deploying multiple countermeasures in the hope of eliminating threats with the understanding that the best that can be done is mitigating part of the risk.

Countermeasures specific to *Screaming Channels* include practices such as masking or hiding, better known as cryptanalytic resistant algorithms. Device manufacturers can use such algorithms in dedicated hardware modules or supplied software libraries. Application developers should ensure that provided algorithms are resistant or implement third-party libraries with these features. Another countermeasure with potentially fewer moving parts is the idea of alternating signal strength, which introduces noise into the signal amplitude making side-channel analysis more complicated without affecting the signal's intended frequency/phase modulation.

Perhaps some of the more effective methods are that of frequent rekeying and the use of frequency hopping; though both introduce their own challenges. For re-keying, NIST provides specific guidance on the cryptoperiod (i.e., lifespan) of a key based on its use and purpose (NIST, 2020). Re-keying in accordance with NIST recommendations, or even more frequently, provides a "moving target" for *Screaming Channels* attacks. Frequency hopping likewise introduces complexity for the side-channel analysis by moving the signal in a pre-planned random pattern across multiple channels. Usable spectrum in frequency hopping must be broad enough to allow for sufficient randomization. While re-keying and frequency hopping provide significant protection, as stated, reliance on any single countermeasure is insufficient. Use defense in depth to provide sufficient protection from *Screaming Channels* attacks.

6. Conclusions

Pretend momentarily that you want to find out if there are mice in your home. If you set a trap and it catches a mouse, you know there was at least one mouse in your home. However, if the trap does not catch a mouse, can you be confident your home is mice-free? Of course not. Similarly, if researchers demonstrate extracting a cryptographic key across a side-channel leak for a particular device, it is clear that the device is susceptible to side-channel attacks. If researchers fail to exploit a target device, though, it does not assure that a leak is absent. It's possible that the researchers were not looking at the correct frequency, using the correct cryptanalysis methods, or aggregating sufficient data. For this reason, it is helpful to use statistics like t-test p-values and signal-to-noise ratios to characterize leaks. Additionally, this article proposes a new metric, leak ratio, that compares the SNR during cryptography to the SNR during non-cryptographic processing. The leak ratio accounts for background SNR and provides a metric as to how significant the leak is against a cryptographic key using a specific leak model. Researchers can use the leak ratio to better characterize potential leaks prior to attempting cryptanalysis to confirm the leak.

This article demonstrated a process to collect and analyze signal traces for *Screaming Channels* leaks. Four target devices were evaluated based on the findings from *Identifying System-on-a-Chip Data Leaks over Radio Transmissions of Small Satellites*. The first device, the nRF52832, was a Bluetooth SoC known to exhibit a leak while the other devices were SoC transceivers that are used or could be used for small satellite communication. Two of the small satellite transceivers, the CC1111 and CC1310, exhibited suspicious signals as highlighted in the prior article. The t-test of these two devices suggested a time slice during which a leak could be present. The final device, the Wio-E5, showed no suspicious signs of a leakage during prior analysis in this series. The leak ratios of all three small satellite transceivers were relatively close to each other and orders of magnitude smaller than the known leak of the nRF52832.

Leak ratio is a proposed measurement for the impact of a leak. These initial findings would suggest that the potential leaks of the three satellite transceivers are less impactful than the leak on the Bluetooth device. However, because the exploitability of the leak was only confirmed on one of the four devices, it is unclear how effective using leak ratio as a measure truly is.

Future efforts against a wider array of targets could help to prove leak ratio as an effective measurement for *Screaming Channels* leaks.

References

1. Backlund, L., Ngo, K., Gärtner, J., & Dubrova, E. (2023, October 4). Secret Key Recovery Attacks on Masked and Shuffled Implementations of CRYSTALS-Kyber and Saber. Applied Cryptography and Network Security Workshops, 13907, 159–177. 10.1007/978-3-031-41181-6_9
2. Bertoni, G. M., & Regazzoni, F. (Eds.). (2021). Constructive Side-Channel Analysis and Secure Design: 11th International Workshop, COSADE 2020, Lugano, Switzerland, April 1–3, 2020, Revised Selected Papers. Springer International Publishing. 10.1007/978-3-030-68773-1
3. Buhan, I., & Mangard, S. (2021, May 7). Computing the Signal-to-Noise Ratio (SNR) for SCA. Ileana Buhan. Retrieved December 1, 2023, from https://ileanabuhan.github.io/general/2021/05/07/SNR-tutorial.html
4. Camurati, G. (2021, April 23). The Screaming Channels project. EURECOM GitHub Pages. Retrieved November 30, 2023, from https://eurecom-s3.github.io/screaming_channels/#Datasets
5. Camurati, G., Poeplau, S., Muench, M., Hayes, T., & Francillon, A. (2018). Screaming Channels: When Electromagnetic Side Channels Meet Radio Transceivers. Proceedings of the 25th ACM conference on Computer and Communications Security, CCS '18(October), 163–177. 10.1145/3243734.3243802
6. Cryptographic Engineering Research Group. (2019). T-test Leakage Assessment — FOBOS User Guide 2.0 documentation. CERG GMU. Retrieved November 1, 2023, from https://cryptography.gmu.edu/documentation/fobos/t_test.html
7. Das, S. (2023, August 9). Mixed Signal Circuit - Definition, Design, Examples. Electronics Tutorial | Best Electronics Tutorial Website. Retrieved November 30, 2023, from http://www.electronicsandyou.com/mixed-signal-circuit-definition-design-examples.html
8. de Micheli, G., & Heninger, N. (2020, December 11). Recovering cryptographic keys from partial information, by example. IACR Cryptology ePrint Archive, 2020(1506), 1-47. https://eprint.iacr.org/2020/1506
9. Dinaburg, A. (2019, November 27). 64 Bits ought to be enough for anybody! Trail of Bits. Retrieved December 1, 2023, from https://blog.trailofbits.com/2019/11/27/64-bits-ought-to-be-enough-for-anybody/
10. European Commission. (2017, July). Digital Transformation Monitor - Low-Earth Orbit satellites: Spectrum access. Advanced Technologies for Industry. Retrieved October 30, 2023, from https://ati.ec.europa.eu/sites/default/files/2020-06/Low-Earth%20Orbit%20satellites%20-%20Spectrum%20access%20%28v1_0%29.pdf
11. Hoofnagle, C. J., & Garfinkel, S. L. (2022). Law and Policy for the Quantum Age. Cambridge University Press.
12. Kolokotronis, N., & Shiaeles, S. (Eds.). (2021). Cyber-Security Threats, Actors, and Dynamic Mitigation. CRC Press/Taylor & Francis Group.

13. Lee, J., & Han, D.-G. (2020, May 16). Security analysis on dummy based side-channel countermeasures—Case study: AES with dummy and shuffling. Applied Soft Computing Journal, 93(2020), 1-9. 10.1016/j.asoc.2020.106352
14. Marshall, A. (2022). Noise Effects in Analog Systems. In Mismatch and Noise in Modern IC Processes (pp. 109-117). Springer International Publishing. https://link.springer.com/chapter/10.1007/978-3-031-79791-0_10
15. NIST. (2020, May). Special Publication 800-57. https://csrc.nist.gov/pubs/sp/800/57/pt1/r5/final
16. NIST. (2023, May 8). Cryptographic Standards and Guidelines | CSRC. NIST Computer Security Resource Center. Retrieved December 6, 2023, from https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines
17. Ouladj, M., & Guilley, S. (2021). Side-Channel Analysis of Embedded Systems: An Efficient Algorithmic Approach. Springer International Publishing.
18. Pearson, K., Griffith, C., Zellem, R., Koskinen, T., & Roudier, G. (2019, January). The American Astronomical Society, find out more The Institute of Physics, find out more Ground-based Spectroscopy of the Exoplanet XO-2b Using a Systematic Wavelength Calibration. The Astronomical Journal, 157(1), 21-40. 10.3847/1538-3881/aaf1ae
19. Randolph, M., & Diehl, W. (2020). Power Side-Channel Attack Analysis: A Review of 20 Years of Study for the Layman. Cryptography, 4(2), 1-33. 10.3390/cryptography4020015
20. Read, C. L. (2022). The Bitcoin Dilemma: Weighing the Economic and Environmental Costs and Benefits. Springer International Publishing.
21. Ryan, M. (2021). Ransomware Revolution: The Rise of a Prodigious Cyber Threat. Springer International Publishing.
22. Schwenk, J. (2022). Guide to Internet Cryptography: Security Protocols and Real-World Attack Implications. Springer International Publishing.
23. Sergienko, I. V., Zadiraka, V. K., & Lytvyn, O. M. (2022). Elements of the General Theory of Optimal Algorithms. Springer International Publishing.
24. Texas Instruments. (2020, June). CC13x0, CC26x0 SimpleLink™ Wireless MCU Technical Reference Manual. Retrieved October 31, 2023, from https://www.ti.com/lit/ug/swcu117i/swcu117i.pdf?ts=1698707923592&ref_url=http%253A%252F%252Fwww.ti.com%252Flit%252Fpdf%252Fswcu117
25. Vaughan, A. (2023, February 6). How to optimize size and power consumption in LEO satellites with FDAs. TI E2E. Retrieved December 1, 2023, from https://e2e.ti.com/blogs_/b/analogwire/posts/size-and-power-in-leo-satellites-with-fdas
26. Wong, D. (2021). Real-World Cryptography. Manning.
27. Zhou, J., Adepu, S., Alcaraz, C., Batina, L., Casalicchio, E., Chattopadhyay, S., Jin, C., Lin, J., Losiouk, E., Majumdar, S., Meng, W., Picek, S., Shao, J., Su, C., Wang, C., Zhauniarovich, Y., & Zonouz, S. (Eds.). (2022). Applied Cryptography and Network Security Workshops: ACNS 2022 Satellite Workshops, AIBlock, AIHWS, AIoTS, CIMSS, Cloud S&P, SCI, SecMT, SiMLA, Rome, Italy, June 20–23, 2022, Proceedings. Springer International Publishing.

Contributions

To perform the analysis described in this article, a significant amount of software was created. Specifically, firmware was created to exercise each device in the described manner. Scripts were written to exercise the device and capture traces using GNU Radio. The original code from the 2018 *Screaming Channels* project was ported from Python2 to Python3 with added functionality to separate the signal capture from the analysis. Multiple Python Jupyter Notebooks were developed to aggregate/align traces and perform the statistical calculations. The Notebooks provide a detailed walkthrough of the analysis including instructions on how to modify configurations for other target devices. All code for the project, step-by-step instructions for reproducing the analysis, and bodies of traces used in the analysis are available at https://github.com/GallagherTom/screaming_satellites.

