

Acoustic Data Exfiltration via Fan Speed Modulation: A Replication Study

Shaurya Chaudhry
shauryachaudhry10@gmail.com

ABSTRACT

Even air-gapped systems can be vulnerable to covert acoustic techniques that exploit physical features of the machine. This paper replicates key elements of the Fansmitter technique on a commonly available Windows desktop. We modulate a single controllable GPU fan between two fixed speed settings, which represent a binary 0 and 1. We recorded the resulting acoustic noise using the “phyphox” app, varying distances between 20 and 50 cm. Then they were analyzed using a Python pipeline, which estimated transmitted bit patterns and computed both binary and confidence-weighted (“soft”) error metrics. Within each window a “closeness” score is computed as $(1 - \text{vote ratio})$, where vote ratio is the expected bit. Results show that fan-speed modulation can produce measurable, decodable structure at close range, but decoding degrades substantially with distance and under realistic background noise. We conclude that the channel is inconsistent beyond short distances in our setup however covert channels can remain partially intelligible.

INTRODUCTION

Air-gapped systems are separated from the internet or other networks (external). This air-gap is achieved by physically disconnecting these systems from other networks (wired and wireless both). This helps prevent communication between these networks/systems. These systems are used in areas where confidentiality and security are necessary to prevent data exfiltration. When not used, systems can be prone to hacker attacks/data exfiltration- which can be seen in the Boeing 787¹.

Our work aims to highlight a technique of exfiltrating data from speakerless airgapped systems through an acoustic covert channel, which is essentially a way of transmitting data discretely that uses sound waves at low frequencies only partially audible to humans, which helps attackers to send their desired information from the targeted system to their own. We will also present a novel way of countering this data exfiltration, making the process less effective, thereby saving data.

In our paper, we are showing a way of exfiltrating data from speakerless air-gapped systems acoustically in an easier manner than the one shown in the paper “Fansmitter” via fan control. Also, we will present a novel countermeasure to make the aforementioned technique of data exfiltration less effective.

This research paper adds to the acoustic covert channel literature by providing a replication of Fansmitter using easily accessible fan control software. We encode binary symbols by maintaining the fan at two constant speeds (25% and 100% in the FanControl app) that produce prominent peaks, record signals with a nearby receiver (or in this case, a phone microphone), and decode the bitstreams.

RELATED WORK

Many different covert channels have been explored in previous papers, exposing the “realism” of data exfiltration from air-gapped systems. There are 4 main types of covert channels, namely electromagnetic, optic, thermal, and acoustic covert channels (the last one is what we will be discussing in this paper). A paper titled “A Survey of Covert Channels and Countermeasures in Computer Network Protocols”⁷ gives a detailed overview of these types of covert channels along with their detection, elimination, and capacity limitation to improve security in future computer networks.

Some work has already been done in the field, which includes the following papers: RAMBO Attack, Diskfiltration, Fansmitter, BRIGHTNESS, etc.^{2,3,4,5} Most of the work in this area has been done by Mordechai Guri. These papers aimed to showcase the vulnerability of data in these systems, even if they’re air-gapped, highlighting data security risks in sensitive areas such as military bases. These papers also provided countermeasures to “diffuse” these cyber attacks and keep these systems’ data safe and secure^{6,7}.

While most of the work in this area has been done by M. Guri, as I’ve already mentioned, the papers titled ‘On Covert Acoustical Mesh Networks in Air’⁶ and ‘A Survey of Covert Channels and Countermeasures in Computer Network Protocols’⁷ also investigate somewhat relevant areas. More specifically, the first paper proposes a covert channel that utilizes audio modulation/demodulation with a network stack that is based on a communication system that is adapted for covert communications by utilizing the ultrasonic frequency range. It also shows how this scenario of covert acoustical communications over the air medium can be extended to multi-hop communications and even to wireless mesh networks. This covert acoustical mesh network is accessible via inaudible audio transmissions. Its applications include the use for remote keylogging over multiple hops, and it also essentially makes many typical security concepts useless, as they don’t consider acoustical communications. It also discusses two countermeasures to the same, namely low-pass filtering in computing systems and a host-based intrusion detection system for analyzing audio input and output to detect any irregularities.

Moving on to the second aforementioned paper, it acts as a survey of all major existing methods for creating covert channels in network and application protocols. It also gives an overview of common methods for their detection, elimination, and capacity limitation, required to improve security in future computer networks. It also discusses four countermeasures, namely eliminating the channel, limiting the bandwidth of the channel, auditing the channel, and documenting the channel.

Electromagnetic Covert Channels

The most extensively explored are electromagnetic emissions, as discussed in a recent paper titled “Air-Gap Electromagnetic Covert Channel” (2023), which exploits electromagnetic emissions from a USB introduced into an air-gapped system, which can be modulated and received by a radio receiver¹⁰. Other papers on the topic include “Soft Tempest: Hidden data transmission using electromagnetic emanations”,⁹ which exploits EM emissions from a video card, AirHopper: Bridging the air-gap between isolated networks and mobile phones using radio frequencies,⁶ which also exploits video card emissions to break air-gaps similarly, “E. Thiele, “Tempest for Eliza,””¹¹ which uses a monitor to transfer radio signals (an electromagnetic covert channel) to a radio receiver, showcasing an important data attack concept which would then later be explored in-depth in various future papers and lastly “GSMem: Data Exfiltration from Air-Gapped Computers over GSM Frequencies,”¹² which exploited EM missions from a motherboard as a covert exfiltration channel.

Optical Covert Channels

Papers that discuss optical covert channels include: “Information leakage from optical emanations,”¹³ which used light from keyboard LEDs, and “Light-based printer attack overcomes air-gapped computer security,”¹⁴ which used malware, remote lasers, and scanners to create an optical covert channel.

Thermal Covert Channels

The thermal covert channel has not been explored as much as the others, but this is one of the prominent papers on the topic- “BitWhisper: Covert Signaling Channel between Air-Gapped Computers using Thermal Manipulations,” in Computer Security Foundations Symposium (CSF), IEEE, 2015”.¹⁴ This paper exploited heat emissions between two neighboring compromised air-gapped computers and the heat sensors of these systems to create a covert communication channel used for data exfiltration.

Acoustic Covert Channels

The acoustic covert channel is the one used and examined in this paper. Two other prominent papers on the topic, besides those already mentioned (such as Fansmitter), are “On Covert Acoustical Mesh Networks in Air.”⁶ and “Bridging the Air Gap: Inaudible Data Exfiltration by Insiders.”¹⁵ The first used audible and inaudible sound for attack, and the second introduced acoustic covert channel attack scenarios with laptops and smartphones.

Research Gap

Across these four categories, the acoustic covert channel is the least dependent on specialised receiving hardware. An electromagnetic channel requires a radio receiver, and a thermal channel requires a second compromised machine placed close to the first, whereas an acoustic channel can in principle be received by any nearby smartphone. Existing acoustic work, however, has generally relied on transmitter software written specifically for the purpose, or has been carried out under controlled acoustic conditions. What has not been established is whether the same channel can be reproduced using only consumer-grade tools — an off-the-shelf fan-control utility on the transmitter, a general-purpose measurement app on the

receiver, and no acoustic isolation of the room. This paper addresses that gap by attempting such a replication and reporting the conditions under which it succeeds or fails.

PROBLEM STATEMENT

We evaluate whether a binary sequence can be transmitted from a speakerless air-gapped desktop to a nearby smartphone using only fan-speed modulation driven by consumer fan-control software, and then recovered offline from the spectral data the phone exports. The transmitter encodes each bit as one of two fixed fan duty cycles held for a fixed interval; the receiver records the resulting sound and exports a peak-frequency time series. The condition varied in the comparison is microphone distance: four distances (20, 30, 40 and 50 cm) are tested, with four distinct 6-bit sequences at each, giving 16 trials and 96 transmitted bits in total. Fixed-speed control recordings at 0%, 25% and 100% duty establish the baseline spectral behaviour of the two states and of the room with the fan idle. A transmission is judged successful to the extent that the decoded sequence matches the transmitted sequence, so bit accuracy is the primary measure; because each symbol is one of two states, 50% accuracy corresponds to chance and is the threshold against which any claim of successful transmission must be assessed. The decoding pipeline itself, and the way its parameters were chosen, are described under Signal Processing and Decoding below.

METHODS

System Overview

The data exfiltration system is centered on a compromised air-gapped computer, which is also known as the transmitter. In this system, the computer's fans are repurposed from simply cooling the system to also playing a role in the intruder's attack to transmit data (assuming the computer is compromised). Essentially, this method establishes an acoustic covert channel through the fan noise for data exfiltration. In our method, we assume the fan speed is adjusted through potential malware being in the air gapped system, which is very possible to do, as demonstrated in previously cited works.^{4,10,12} The infected computer would then collect the attackers' desired sensitive data, such as passwords, and modulate it over the acoustic soundwave signals from the computer's fans. To encode the data over the fan's acoustic signals, the data must be modulated and in this paper binary data will be modulated over the fan sound signals, which then transmit to a nearby receiver which can be as simple as a phone with a microphone picks up these signals and this can be then demodulated by the intruders (creating a one-way communication link between the air gapped systems' fans and the intruder's phone). The whole reason this setup works in air-gap computers is that air and sound are the medium of transfer and not online networks, which is why the air-gaps can be breached, as they were designed mainly against internet-based cyber attacks.

Experimental Setup

We used the “FanControl” app to control the fan speeds.¹⁶ We controlled it in terms of percentage, increasing 50% per second when going from 25 to 100 in 2 seconds. The two percentages of fan speed taken were 25% and 100% fan speed, and the software didn’t mention the RPM, but there was a significant difference in sound between the 2 percentages. An iPhone 16 smartphone microphone was used to capture the sound data. The application used was phyphox, particularly the audio spectrum feature to find the Fourier transform.¹⁷ After each trial, we moved the phone 10cm away from the computer for the next one to see at what point the distance of the phone from the computer made a significant difference to the binary data modulated. We predicted that the further it goes, the less accurate it would be due to the lesser proximity to the computer’s fans’ signals. It was in a straight line away from the fan; we didn’t test the angles, and the microphone was oriented towards the computer. The environment was quite noisy, but still realistic, to be frank, as it was in a shared space of people working on projects such as this one. No significant steps were taken to minimize background noise. Still, we didn’t feel this should matter significantly, as the signals of the acoustic soundwaves from the computers’ fans would be distinct from the rest of the background noise, allowing us to tell which one the computers’ signals are looking at by examining the Fourier transform. The distances taken between the transmitter and the receiver were 20cm, 30cm, 40 cm, and 50cm, and we used 4 distinct bit strings for each distance, too.

Data Collection Procedure

We used specific 6-digit binary data sequences to exfiltrate the data. This binary data would be encoded by the fan’s speed modulation, which has been discussed. Each recording was 30 seconds, and every 5 seconds, I would increase or decrease the fan speed from 25% to 100%, depending on the sequence, i.e., if it started with “10,” I would start with 100% and then decrease it to 25%. We intended that 100% fan speed would be decoded as ‘1’ in binary and 25% as ‘0’ in binary. There was a slight issue, though, as the fan would take 2 seconds to reach the intended value, as the maximum the software allowed me to increase/decrease the fan speed was 50% per second, i.e., it would go from 25% to 75% to 100% in 2 seconds, leading to a slight delay in the timings taken. The number 5 seconds was taken to ensure sufficient time for the microphone to decipher each bit. More specifically, the fan inertia and need for sound stability led to a relatively long bit rate and hence a lower data rate. There would obviously be a short gap between me increasing or decreasing the fan speed as well due to uncertainty, too. A 5-second delay was implemented before starting the 30-second timer to reduce uncertainty, too. As the fan speeds were modulated, the microphone obviously kept on recording this sound to pick up the binary data, and the format was .csv. There were different conditions, though, such as an increase in distance by 10cm to observe the differences in the accuracy of the data collected as the distance between the computer and phone microphone increases.

Signal Processing and Decoding

Figure 1. Peak-frequency trace from a single transmission run. The trace alternates between the two fan-state plateaus (around 47 Hz at 25% duty and around 70 Hz at 100% duty), demonstrating that the FanControl-driven bit modulation is cleanly recoverable as a peak-frequency time series.

Using the FanControl application, we identified a GPU fan that we could control in our system. We set the fan speed of that fan at various fixed percent values, for example, 25% and 100%. We used these values as a baseline for frequency identification. When the fan was set to 25%, we would take that to mean bit 0, and when the fan was set to 100%, we took that to mean bit 1. We made sure no other software was using the GPU fan or changing its speed during the experiment.

Control recordings were taken with the fan held at 0%, 25% and 100% duty in order to establish the baseline spectral behaviour of each state. The resulting peak-frequency distributions and the decoding boundary derived from them are shown in Figure 2. (A passage previously appearing here described peak activity in the 300–700 Hz region; it referred to a figure not included in this paper and was inconsistent with the measured 47 Hz / 70 Hz mapping used throughout, and has been removed.)

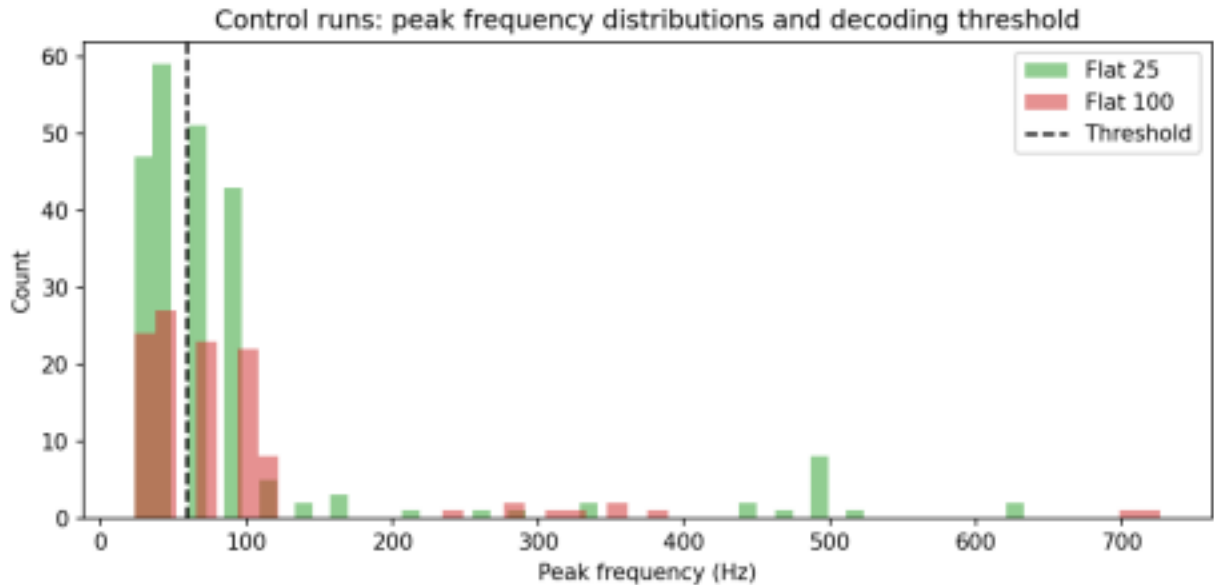


Figure 2. Control runs: peak-frequency distributions and decoding threshold. The two fan-state distributions separate around a midpoint threshold near 58 Hz, supporting the use of a single fixed decoding boundary between the 25% and 100% states.

Our method uses binary frequency-shift keying (BFSK), in which each of the two symbols is carried by a distinct tone produced by holding the fan at a fixed speed plateau. A 0 is sent with the fan at 25% duty and a 1 with the fan at 100% duty, the 100% state producing the higher tone. This terminology is used consistently throughout the paper. The FanControl app was instructed to increase the fan speed from 25% to 100% at a rate of increase of 50% per second, or 2 seconds to change fan speeds from 25 to 100% and vice versa. These values were chosen to produce clearly distinct frequencies, which have been discussed previously. An earlier draft of this paper quoted nominal RPM-derived estimates of approximately 67 Hz and 107 Hz for the two states. Those figures were estimates from nominal fan speeds and were not the values observed in the recordings; they have been removed. Throughout this paper the two states are described by their measured peak frequencies in the control recordings: approximately 47 Hz for the 25% state (“0”) and approximately 70 Hz for the 100% state (“1”), with a decoding boundary near 58 Hz. Each bit signal lasted 5 seconds before being changed, as there were 6 bits in each pattern chosen and hence 30 seconds overall. At the end of the 5 seconds, I changed the speed on the FanControl app, which would have an effect after 2 seconds of the FanControl App increasing the speed by 75% or reducing it by 75%, creating distinct binary states. We also had 5 seconds before starting the 30-second timer to minimize initial uncertainty. A longer bit period is likely to be more reliable for data collection, but the con for this is that the data rate of transmission would be even slower, which is quite a common occurrence in the acoustic covert channel. Only one fan was able to be controlled properly by the FanControl app, as it was the only one where there was an audible difference in the sound of the fan after changing the fan speed percentages. We were unable to use multiple fans as a result.

Each message was composed of 6 bits, and there was no specific known preamble sequence before the actual data payload. To replace this, we decided to use a 5-second preamble timer before starting the actual timer to reduce initial uncertainty. However, if we did use a preamble sequence, it would help the receiver/microphone detect the start of a transmission and synchronize the bit timing while also providing the receiver with a “standing out” signal in the processed signal or a distinct audio signature. This addition could be a useful addition to improve the reliability of the experiment. The 5-second preamble timer allowed us to decrease timing uncertainty and also set the fan speed to the corresponding state of the first bit of the given bit pattern, which helps increase the accuracy of data transmission.

After this preamble time, the actual desired data bits for the attacker start to transmit. This would be especially useful for small pieces of data like passwords due to the low transmission rate of the channel. Due to this slow speed of the channel, only the critical information is sent. Due to the change from one bit to another taking 2 seconds in the FanControl app, at the end of the 5 seconds of each bit, there would be a short gap to separate repeating transmissions, for example, a 1 followed by another 1. Note that, for the first bit right after the preamble, there was no such 2-second gap, as this was already set up according to the first bit state in the 5-second preamble time period. This gap, as a result, helped prevent overlapping frames. We repeated the procedure for multiple distances and bit patterns, which will be discussed elsewhere, to enhance the reliability of the experiment and also to understand the extent to which the distance of the receiver from the speaker affects the accuracy of the binary bits decoded.

After capturing the audio, the receiver must process the raw signal to extract the useful data via signal processing techniques such as Fourier analysis. These techniques are used to detect the fan’s speed changes in the recording and hence interpret the data (0’s and 1’s).

In the “phyphox” app on the phone (receiver). We chose the audio spectrum option in this app, which allowed us to analyze the Fourier transform of each binary bit pattern, and then we extracted the data in .csv format. The Fourier transform is essentially just a technique used to decompose a signal into its fundamental frequencies, specifically, how much of each frequency is detected, from which we could detect the frequencies that correlate to the frequencies of the 0 and 1 binary states of the fans or 25% and 100% speed, respectively. Given the earlier analysis, if the fan frequencies are around 40–120 Hz, for example, the Fourier transform allows us to focus on those specific frequencies in that range and differentiate them from frequencies outside that range. This technique, hence, allowed us to eliminate environmental sounds that are not in the expected range of the fan’s noise.

This processed data is then analyzed to see the prominent frequency changes. We used the aforementioned Fourier transform to create a spectrogram graph, which essentially allowed us to compute the frequency spectrum for each overlapping time window (which the graph helped create by splitting up the audio recording into these windows), showing the peak frequency at each moment in the audio recording. This allowed us to understand the shifts between the two prominent frequencies visually, but for deeper analysis, we used Python code to observe them. This program allowed us to essentially scan the spectrum of a particular time segment to find the frequency of the maximum power, which should logically be the fan’s tone if the signal is clear, and hence when you increase the distance of

the receiver from the transmitter, the signal will become less and less clear, and hence, the interpretation of the data will also become less accurate. The overall result of this procedure is just a list of the peak frequency values or, ideally, the fan's tone indexed by time, converting the raw audio data extracted from the app into a clear representation of the peak frequency vs time curve, showing the shifts of the most prominent fan tone through the course of the transmission. The data of this curve needs to be aligned with the known bit duration of the particular bit string. The receiver knows how long each bit lasts (which has been discussed in previous sections), allowing the analysis to decompose the peak frequency timeline into individual bits.

Parameter Selection

The decoding pipeline has three free parameters: a decision threshold applied to the peak-frequency series, a start offset locating the beginning of the six-bit frame, and a stretch factor scaling the assumed frame duration. The offset and stretch factor are required because the fan transitions were triggered by hand rather than by software, so the exact start time and pacing of each transmission are not known in advance. All three were selected by exhaustive grid search. Candidate thresholds were drawn from the 30th to the 70th percentile of each recording's in-band (20–300 Hz) peak-frequency distribution in steps of five percentiles; candidate stretch factors ran from 0.85 to 1.15 in steps of 0.05; and for each stretch factor, nine candidate start offsets were tested, spanning zero to one window length. This gives 567 parameter combinations per recording. For each combination the peak-frequency trace was divided into six equal windows, and a vote ratio — the fraction of samples in that window exceeding the threshold — was computed for each. The combination retained was the one that maximised mean closeness to the transmitted bit sequence, where closeness is defined as the vote ratio for windows whose transmitted bit is 1 and one minus the vote ratio for windows whose transmitted bit is 0. Decoded bits were then assigned by thresholding each window's vote ratio at 0.5, and the soft-error rate reported below is $100 \times (1 - \text{mean closeness})$ under the retained combination. The selected threshold was therefore not constant across recordings: values of 23.4 Hz, 46.9 Hz and 70.3 Hz were retained for different runs.

RESULTS AND DISCUSSION

This section presents the key outcomes of the experiments. A total of 16 trials were conducted at four microphone distances—20 cm, 30 cm, 40 cm, and 50 cm—relative to the fan exhaust.

Spectral Separability

Spectral separability refers to how clearly the two fan states (bit 0 vs bit 1) can be distinguished in frequency space.

Across runs, the exported peak-frequency traces (Figure 1) and spectrograms showed two recurring frequency bands associated with the two fixed fan duty-cycle settings. In our setup, the low-duty state (25%) typically clustered around ~47 Hz, while the high-duty state (100%) clustered around ~70 Hz. These clusters were not perfectly clean: the peak tracker occasionally jumped to higher harmonics or noise-related peaks, especially in louder background conditions. However, the repeated reappearance of

two dominant bands provided measurable structure that could be used for decoding. These baseline clusters and the decoding threshold are summarized in Figure 2.

Rather than relying on perfectly timed 5-second boundaries, the analysis treated timing as uncertain (because fan transitions were not instantaneous and were manually triggered). The decoding pipeline, therefore, used the peak-frequency time series to evaluate candidate segmentations by sweeping over small timing offsets and slight stretch factors, and it selected thresholds from quantiles of the in-band peak-frequency distribution. This best-fit approach exploits the fact that, when the two fan states are spectrally separable, the fraction of samples above a threshold (“vote ratio”) changes systematically between low and high states.

Figure 3 illustrates this effect by comparing an example near-distance run (where the two frequency bands are more consistently separated and the decoded bit pattern largely matches the transmitted pattern) to a far-distance run (where spectral overlap and peak-tracking instability increase and the decoded pattern degrades).

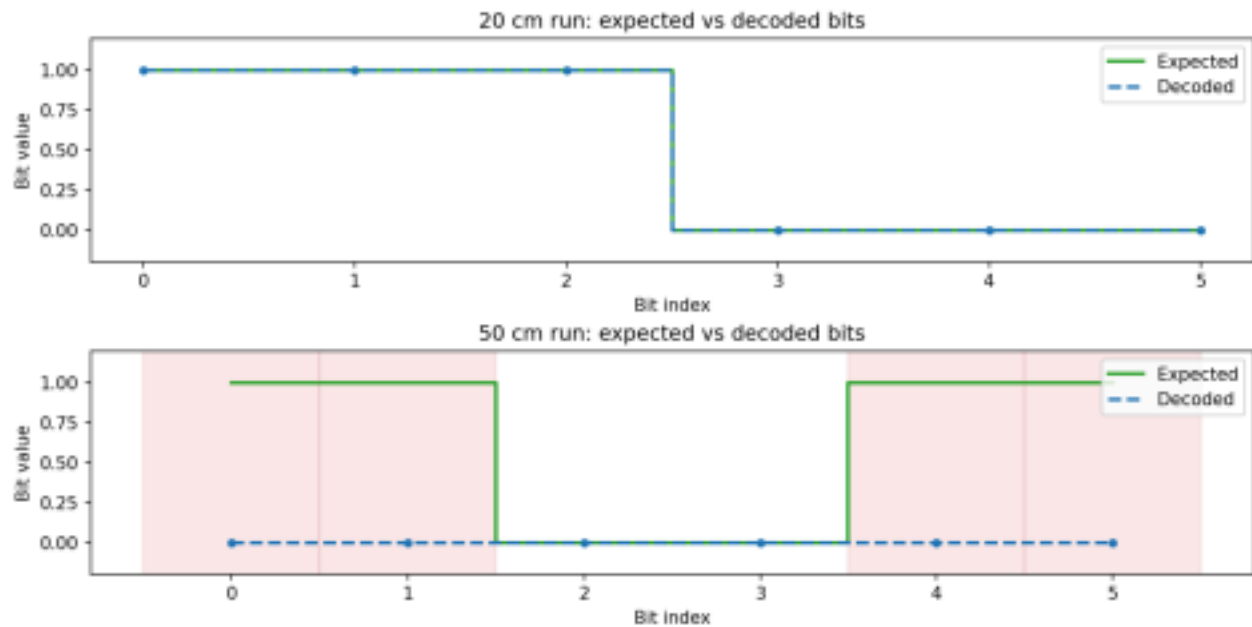


Figure 3. Expected vs decoded bit patterns at near and far distances. At the closer distance, the decoded pattern closely tracks the transmitted bits, while at the far distance, spectral overlap and peak-tracking instability cause repeated bit flips — showing that distance, not encoding, dominates the error budget.

Performance

Bit patterns such as ‘011010’ were transmitted using preconfigured high and low fan states and decoded from exported peak-frequency data using an offline analysis pipeline. We report two error measures:

Oct 2026

Vol 12, No 2.

binary error and soft error.

The Python pipeline loads each run's "Peak History.csv", which is a time series of dominant (peak) frequency values. As there is a potential that the fan transitions are not perfectly aligned to the intended timing, the script has to search over the entire time-series data. We use a "best-fit procedure" that searches over: a decision threshold, which is picked from quantiles of the in-band peak-frequency distribution; a small time offset, which can be varied; and a stretch factor that slightly rescales the window duration we are exploring. We sweep over the settings and then divide the recording into six windows, and then find a "vote ratio" over the window. This is the fraction of samples whose peak frequency exceeds a certain candidate threshold.

The binary decoded bit for each window is then a hard decision: if the vote ratio is greater than 0.5, the bit is decoded as 1; otherwise, it is decoded as 0. The binary error rate is the percentage of bits where this decoded bit differs from the transmitted (expected) bit.

The soft metric uses the same vote ratios but scores them continuously instead of forcing an all-or-nothing decision. For each window, a "closeness" score is computed as the vote ratio when the expected bit is 1, or $(1 - \text{vote ratio})$ when the expected bit is 0. The soft error rate is then calculated as $100 \times (1 - \text{mean closeness})$. In other words, soft error penalizes confident wrong windows more than borderline windows, and it gives partial credit when the vote ratio is near the correct side of the threshold, even if the hard decision flips.

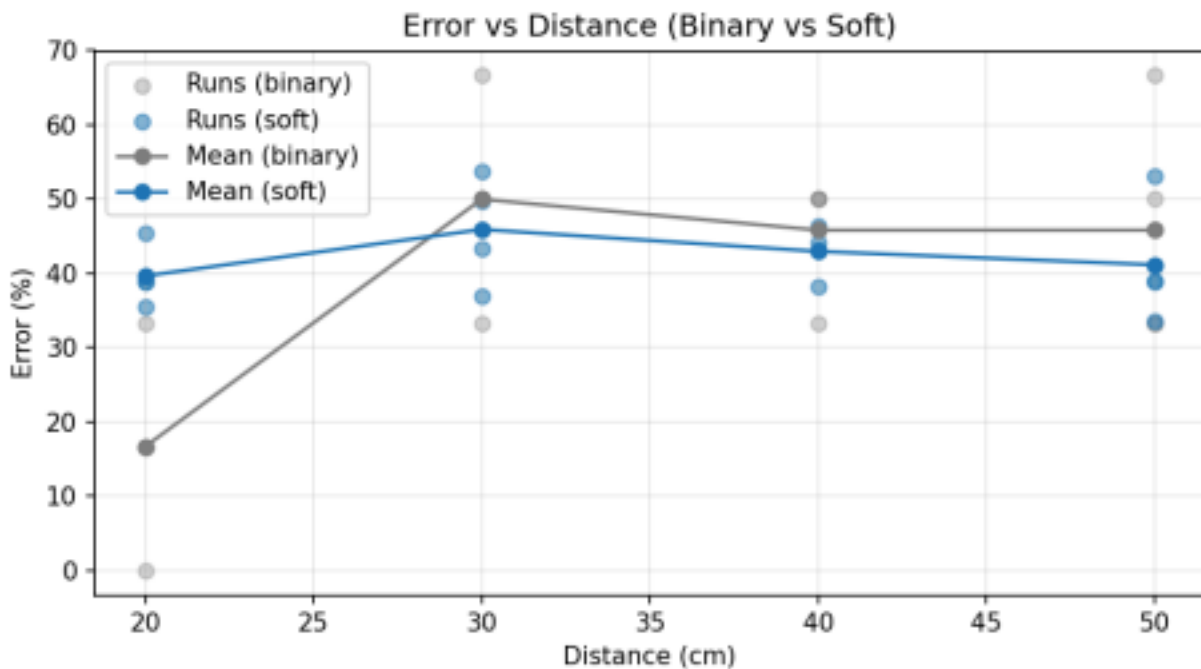


Figure 4. Error vs distance for binary vs soft scoring. Both error curves grow steadily with distance, but the soft-error curve sits a few percentage points below the binary-error curve at every range, indicating

that the channel often produces partially-correct decodings even when full bit recovery fails.

Looking at Figure 4, soft error rates are consistently a few percentage points lower than binary errors across most distances. This means that even when the system makes decoding mistakes, it often gets “close” to the correct bit. At 20 cm, for instance, although some binary errors occur, the soft scores suggest those mistakes were borderline. This gap also confirms that the decoding algorithm’s confidence degrades gradually—not abruptly—as distance increases. It also shows that even at longer distances, the signal still carries some useful structure.

Interpreted practically: the covert channel remains partially intelligible even when full bit recovery fails. For defensive systems, this suggests countermeasures must degrade not just decoding success but signal clarity itself. For attackers, it means the method might still extract partial secrets in imperfect conditions.

CONCLUSION

This replication study shows that fan-speed modulation on a commodity Windows desktop can produce a measurable acoustic signature that carries decodable structure, but only under short-range conditions in our setup.

Using two fixed fan duty-cycle states and an offline peak-frequency decoding pipeline, we were able to recover some 6-bit patterns at close proximity; however, decoding performance degraded substantially as microphone distance increased and as peak-tracking became less stable in a realistic, noisy environment. Beyond short distances, the channel became unreliable for consistent bit recovery.

These results highlight several practical constraints that limit the real-world usefulness of the channel: the bandwidth is inherently low (with ~ 5 s per bit plus transition time, the throughput is only about 0.1–0.15 bits/s, so only very small payloads are realistic), performance is strongly distance-sensitive (our error rates worsened substantially as the microphone moved farther away, implying the receiver must be placed close to the system), environmental variability matters (background noise and peak-frequency tracking instability reduced spectral separability and directly hurt decoding), and the modulation itself may be detectable (switching between two fan states can be audible in quiet settings, increasing the chance of human notice).

Despite these limitations, the study reinforces a broader security point: even without speakers or network connectivity, physical side effects can create unintended communication paths. As a low-cost mitigation, defenders can reduce channel reliability by introducing controlled randomness into fan control (e.g., small PWM jitter or brief, irregular speed perturbations), which would blur spectral separation and weaken timing consistency without requiring new hardware. Future work should evaluate such countermeasures experimentally and test more robust decoding methods under varied acoustic conditions.

ACKNOWLEDGEMENTS

I appreciate my family's support through the course of this research paper as well as the supportive learning environment at my school.

REFERENCES

1. Zetter, K. FAA: Boeing's New 787 May Be Vulnerable to Hacker Attack. *Wired*, Jan 4, 2008. <https://www.wired.com/2008/01/dreamliner-security/> (accessed 2026-01-06).
2. Guri, M. RAMBO: Leaking Secrets from Air-Gap Computers by Spelling Covert Radio Signals from Computer RAM. *arXiv* 2024, arXiv:2409.02292.
3. Guri, M.; Solewicz, Y.; Daidakulov, A.; Elovici, Y. DiskFiltration: Data Exfiltration from Speakerless Air-Gapped Computers via Covert Hard-Drive Noise. *arXiv* 2016, arXiv:1608.03431.
4. Guri, M.; Solewicz, Y.; Daidakulov, A.; Elovici, Y. Fansmitter: Acoustic Data Exfiltration from (Speakerless) Air-Gapped Computers. *Computers & Security* 2020, 91, 101721. <https://doi.org/10.1016/j.cose.2020.101721>.
5. Guri, M.; Bykhovsky, D.; Elovici, Y. BRIGHTNESS: Leaking Sensitive Data from Air-Gapped Workstations via Screen Brightness. *arXiv* 2020, arXiv:2002.01078.
6. Hanspach, M.; Goetz, M. On Covert Acoustical Mesh Networks in Air. *Journal of Communications* 2013, 8 (11), 758–767.
7. Zander, S.; Armitage, G.; Branch, P. A Survey of Covert Channels and Countermeasures in Computer Network Protocols. *IEEE Communications Surveys & Tutorials* 2007, 9 (3), 44–57. <https://doi.org/10.1109/COMST.2007.4317620>.
8. Guri, M.; Kachlon, A.; Elovici, Y. USBee: Air-Gap Covert-Channel via Electromagnetic Emission from USB. *arXiv* 2016, arXiv:1608.08397.
9. Kuhn, M. G.; Anderson, R. J. Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations. In *Information Hiding*; Springer, 1998; pp 124–142. https://doi.org/10.1007/3-540-49380-8_10.
10. Guri, M. Air-Gap Electromagnetic Covert Channel. *IEEE Transactions on Dependable and Secure Computing* 2024, 21 (4), 2127–2144. <https://doi.org/10.1109/TDSC.2023.3300035>.
11. Thiele, E. Tempest for Eliza. <http://www.erikyyy.de/tempest/> (accessed 2026-01-06).
12. Guri, M.; Kedma, G.; Kachlon, A.; Elovici, Y. GSMem: Data Exfiltration from Air-Gapped Computers over GSM Frequencies. In *24th USENIX Security Symposium (USENIX Security 15)*, USENIX Association: Washington, DC, 2015.

13. Loughry, J.; Umphress, D. A. Information Leakage from Optical Emanations. *ACM Transactions on Information and System Security* 2002, 5 (3), 262–289.
14. Guri, M.; Bykhovsky, D.; Elovici, Y. BitWhisper: Covert Signaling Channel between Air-Gapped Computers Using Thermal Manipulations. In 2015, IEEE 28th Computer Security Foundations Symposium (CSF), IEEE, 2015.
15. O'Malley, S.; Choo, K.-K. R. Bridging the Air Gap: Inaudible Data Exfiltration by Insiders. In *Proceedings of the 20th Americas Conference on Information Systems (AMCIS 2014)*; Association for Information Systems: Savannah, GA, 2014.
16. Rem0o. FanControl (Releases). GitHub repository.
<https://github.com/Rem0o/FanControl.Releases> (accessed 2026-01-06).
17. Staacks, S.; Heinke, H.; Stampfer, C.; Horn, M. Advanced Tools for Smartphone-Based Experiments: phyphox. *Physics Education* 2018, 53 (4), 045009.
<https://doi.org/10.1088/1361-6552/aac05e>.

ABOUT THE AUTHOR

Shaurya Chaudhry is currently in grade 10 studying at UWCSEA (Dover Campus) in Singapore. His interests include mathematics, physics, and cybersecurity. He wants to be involved in a STEM field in his future studies. In addition to his academic interests, he enjoys playing chess too.