CYSE 587 (Spring 2025)

Cyber Security System Engineering

Lab1: ADS-B Signal Spoofing and Jamming

Team Securetight

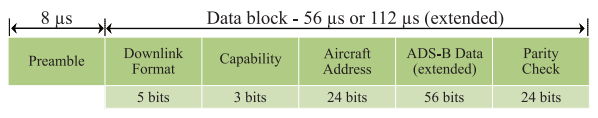
1. Tasks to perform

ADS-B spoofing involves broadcasting fake aircraft location signals to deceive aviation systems. Attackers can create nonexistent aircraft or deceive real aircraft positions, potentially causing dangerous confusion in air traffic management and increasing collision risks.

The task is to enhance the channel to be more realistic and implement gradual spoofing attacks and various jamming attacks in the given simulation source code.

1. Adding realistic features to the ADS-B channel implementation

Before making some modifications to the spoofing and jamming, we need to first change the way the ADS-B channel works in the simulation. We introduced the format of actual ADS-B messages to perform spoofing and jamming attacks for future use while adding the bit-level transmitting and receiving capabilities and timing concepts of the channel. We boldly removed the artificial "corrupt" message implementation and used the 3-byte parity of the message to perform a cyclic redundancy check(CRC).



*Figure 1. [[1]](#endnote-1)Illustration of ADS-B message structure*

We implemented the whole ADS-B message format. However, the preamble signal is not transmitted by the drones in our implementation. Rather, we just added 8μs delay to all the messages assuming that the preamble signal is always correctly transceived. While we acknowledge that distortion of the preamble signal is not possible in our simulator (and some jammers actually target the preamble signal), we believe this is enough to perform a gradual spoofing attack and all the requested jamming attacks.

Messages have a fixed Downlink Format value of 17, Capability 5, and Type Code 11. Drones fit their unique identification(ICAO 24, displayed as Aircraft Address in Figure 1) and the current location in the ADS-B message format, encode it into a hexadecimal string, and then broadcast the signal. For Compact Position Reporting (CPR), drones in our simulation transmit even and odd CPR messages simultaneously, simplifying the location update process.

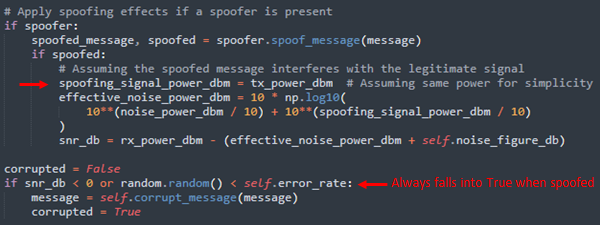
Note: Our implementation assumes an 8μs delay for all transmitted messages, modeling the preamble signal while acknowledging that real-world jammers often target the preamble.

As for jammer implementation, ADS-B messages are encoded into a hexadecimal string and transmitted bit-by-bit. Simulated jammers are capable of emitting a signal with a certain power and frequency, at the desired timing. This signal will flip one bit in a message based on probability. On the other hand, a spoofer does not need a bit-by-bit transceiving concept. Thus, our spoofer implementation receives the full message, decodes the payload within, modifies the data, encodes it again with the correct parity, and then finally transmits it; hoping for GCS to take it as a benign signal.

1. Implementing Gradual Spoofing Attack

A gradual spoofing attack involves slowly and incrementally modifying aircraft position data over time, making the false data harder to detect compared to sudden position changes. The attacker gradually shifts reported coordinates to make the deception appear more natural and bypass anomaly detection.

We had to focus on two main features to implement a realistic gradual spoofing attack. One is a spoofing signal power and the other is a perturbation of a drone position. We will first argue and explain what we have modified to be more realistic regarding spoofing signal power.



*Figure 2. Spoofing signal power setup in the original implementation*

Setting spoofing signal power as the same as the signal power of the message itself is a bad approach regarding “realistic” simulation because spoofers in the real world typically use lower power to avoid detection, i.e., having equal power would make the attack too obvious. Also, if the signal power for a spoofer is too big, the signal-to-noise ratio(*snr\_db*) will fall into a negative value, causing every spoofing message to be classified as corrupted. This should be avoided. So we decided to carefully calculate spoofing\_signal\_power\_dbm in such a way that it does not overwhelm legitimate signals but is strong enough to be injected. In other words, we need to find the value of “spoofing noise” that satisfies the following inequality.

*transmit power – path loss – thermal noise – default noise – spoofing noise = receive power(SNR) > 0*

We targeted the *receive power* to be 30dB (fixed) to get a proper spoofing noise power. This assumes that the attacker knows the transmission power of the drone, whether by calculating it or through preliminary investigation. Message will never be "corrupted" by the spoofer because the attacker targets the SNR value to stay in positive range and re-calculates parity bits after position was spoofed. However, the message still can get corrupted by natural error(displayed as *error\_rate*) and when used with jammer.

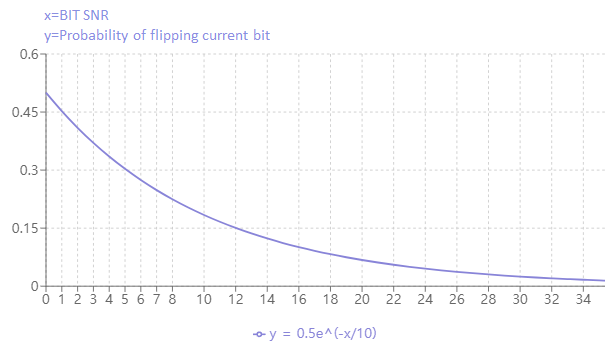
For the perturbation of a drone position, we introduced the concept of acceleration and decay factor λ. Our gradual spoofer now stores previous the position of a drone and calculates its direction vector based on the difference in the drone’s position.

*Figure 3. Formula for updating position data with acceleration vector*

Finally, we calculate the desired spoofing position by adding a Δ vector to the current drone position using the formula(Figure 3). Performing gradual spoofing in this way has several advantages for the attacker. Accumulation of the Δ vector calculated by the direction vector and acceleration helps the spoofing attack to be “gradual” in the concept of momentum. Momentum can increase or decrease in a desired way when λ is set to be greater than 1.0 or less than 1.0, respectively.

1. Implementing Jamming Attacks

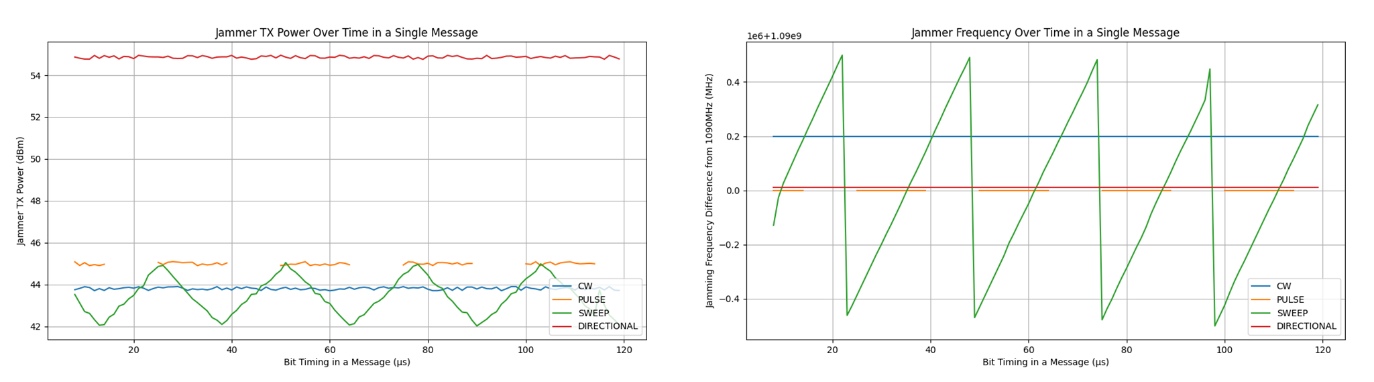
As stated previously, we implemented jammer that is capable of emitting a signal with desirable power, frequency, and timing. In terms of timing, we simulated bit-by-bit transmission for realistic jamming. We first get the jamming signal power for whichever jammer used, combine them together with current SNR in bit-divided time sequence, and then calculate probability for bit to be flipped or not. We set our maximum probability to be 50% on SNR=0 (Figure 4). This means that the more jamming power, the more likely a bit will be flipped. If a jammer successfully flip only one bit, it will eventually be classified as a corrupted message by parity bit calculation.



*Figure 4. Exponential decay function for getting probability of flipping a bit*

All of the jammer that we implemented follows the fact that jamming effectiveness decrease at higher frequencies. This is due to many reasons; e.g., higher frequencies experience more path loss, more affected by atmospheric conditions, etc. We designed the Jammer to suffer a power loss of up to maximum 3 dB due to this phenomenon, and limited the maximum difference from the center frequency up to 500 kHz. Difference higher than 500kHz would be too easy for jamming filter to remove this signal out. That is, a power loss of 3 dB occurs at a frequency difference of 500 kHz. This is a loss of about 1 dB per 167 kHz, implemented linearly.

Now that our simulator has ability to corrupt the message with signal interference, we can implement jammers. We implemented four types of jammer; Continuous Wave(CW), Pulsed, Sweeping, and Directional.



*Figure 5. Jammer TX power (left) and jammer frequency(right) over time for single message*

For CW jammer, the goal is to maximize the interference with the target signal while minimizing the ability of the receiving system to easily filter out the jamming signal by placing it slightly off the target signal's frequency, making it harder to remove through simple filtering techniques. We added 200kHz from ADB-S’s center frequency to do this, avoiding frequency difference exceeding 500kHz. This causes a little bit of power reduction for CW jammer.

For pulse jammer, we implemented the jammer to repeatedly turn on and off the signal emission. Pulse repetition shown in Figure 5 is actually too much in reality. It normally targets only few bits to be flipped to retain its’ stealthiness. But we set the repetition to be much faster than reality to display that our jammer is configurable.

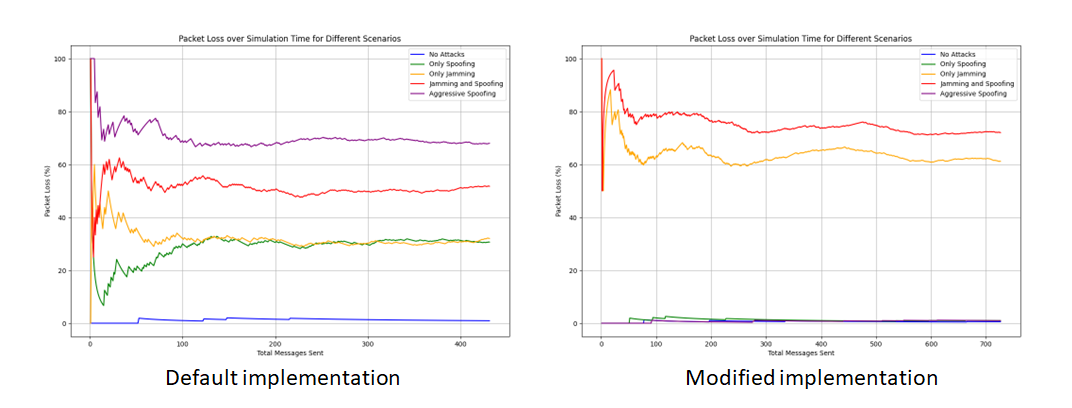
The goal of sweep jammer is to impair a broader communication band by "sweeping" through different frequencies with its full power. One can confirm that our sweep jammer “sweeps” its’ frequency over time from Figure 5. It was interesting to find out that the jamming power also oscillates, thanks to implementation of power reduction related to frequency.

A directional(Beamforming) jammer is built to intentionally disrupt a target communication channel by directing a concentrated, focused jamming signal towards the intended recipient using multiple antennas arranged in an array. Although this simulator does not implement multiple antennas to receive the signal in GCS, we can still simulate directional jamming. Assuming that the attacker is setting the antenna orientation by eye measurement, we introduced a concept called “uncertainity” which the direction towards the GCS from the jammer is slightly different from the exact azimuth. However, since the jammer can form a beam, the difference in azimuth can be overcome if the beam width can cover it. We made the antenna gain value of the jammer to be reduced as the GCS is far from the center of the beam, decreasing the possibility for a bit to be flipped. While the gain is not actually "reduced" in reality, this implementation reflects the importance of aligning the jammer’s antenna direction towards the GCS.

1. Improvements, Achieved Results, and Conclusion

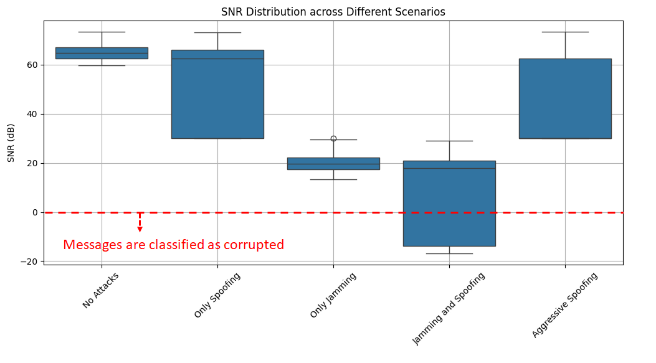
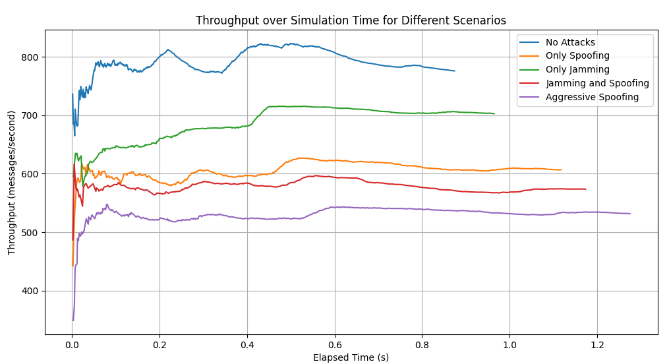
First, there was a distinct difference in the packet loss graph over time compared to the default implementation. We now can confirm that spoofing does not corrupt messages anymore. We also achieved more packet loss for the jammer(pulse jammer was used in the evaluation) as expected. Of course, you can always adjust the pulse width or jamming power to produce higher packet loss. But it would make the attack obvious and will be easily detected by anomaly detection in reality.

If an attacker uses a jammer alongside with a spoofer, messages are more likely to be corrupted because the attacker is now emitting two signals simultaneously to interfere with. Doing so may lower down SNR value towards negative, resulting messages to corrupt easier.

**

*Figure 6. Packet loss over time graph comparison between two implementations*

Simultaneous interference of spoofing and jamming signals causing message to corrupt can also be confirmed in SNR distribution graph(Figure 7). In scenario which the attacker uses jammer and spoofer together tends SNR value to be pushed under 0.

*Figure 7. SNR distribution across different scenarios Figure 8. Throughput over simulation time*

Something interesting was spotted in the throughput evaluation(Figure 8). The aggressive spoofing scenario showed the worst throughput, which there shouldn’t have been a difference compared to “Only Spoofing” scenario programmatically. After some investigation, it turns out that the more aggressive the spoofer is, the more likely it needs expensive calculations; that is, decoding the message, calculating the spoofing position, and encoding the message again, which eventually resulting a spoofer to be more delayed even if there was no explicit delay implementation. It acts just like how the actual delay is generated for the “real” spoofer in the wild.

This exercise directly helps us understanding the importance of building up the security models against cyber physical system. Unlike “security by obscurity”, the drones were communicating with a public-known channel and format which can be easily targeted by threat actors anytime. We can see with our own eyes how actual attacks are done under certain circumstances. By implementing realistic attacks and analyzing their impact, we gained critical insights into securing CPS infrastructure against spoofing and jamming threats.

1. X. Ying et al, "Detecting ADS-B Spoofing Attacks using Deep Neural Networks", arXiv:1904.09969 [cs.CR], Apr. 2019., Fig 2. [↑](#endnote-ref-1)