

PAVE-MAVLink: Formal Verification of MAVLink 2 for Secure UAV Communications

Tom Wray

Stevens Institute of Technology, Hoboken, NJ
thomas.wray@L3harris.com

Ying Wang

Stevens Institute of Technology, Hoboken, NJ
ywang6@stevens.edu

Abstract—This work presents a multi-stage, Large Language Model(LLM)-accelerated framework for formal verification and security validation of UAV communication protocols, focusing on MAVLink 2. We introduce a prompt-driven approach to automatically generate symbolic models for nuXmv and ProVerif, allowing rapid identification and mitigation of protocol vulnerabilities. Experimental results demonstrate that, without cryptographic protections, MAVLink 2 is susceptible to critical command injection attacks, validated through symbolic model checking, software-defined drone simulation, and over-the-air (OTA) testing on physical UAVs. Incorporating ChaCha20 based encryption eliminates these vulnerabilities, as confirmed by formal analysis and empirical validation. These findings illustrate the effectiveness and flexibility of LLM assisted workflows like Prompt Aided Formal Verification (PAVE) while providing a platform for robust cryptographic extensions in advancing UAV protocol security.

Index Terms—Drone, Formal Verification, MAVLink, Prompt Engineering, Security

I. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are increasingly deployed across civilian and defense sectors for tasks such as logistics, surveillance, and autonomous operations. As their autonomy and ubiquity grow, so does the critical need for secure and trustworthy communication protocols. Vulnerabilities in command and control (C2) channels, telemetry links, and protocol logic can be exploited through spoofing, injection, replay, and denial-of-service attacks, posing significant risks to flight safety and mission assurance [1]. Ensuring the correctness and robustness of these systems requires rigorous verification techniques that extend beyond simulation and empirical testing. Formal verification, which uses symbolic models to mathematically prove or refute system properties, has long been a trusted approach for protocol assurance. However, its practical application remains hindered by the need for manual modeling, tool-specific syntax, and expert intervention. Recent advances in Large Language Models (LLMs) offer a transformative opportunity to streamline this process. By leveraging prompt-driven generation and reasoning capabilities, LLMs can translate natural-language specifications into formal models, temporal logic properties, and symbolic representations suitable for model checkers and protocol analyzers [2]–[4].

Fully autonomous eVTOL systems, which operate without onboard pilots require real-time command and control (C2) to

maintain safe and resilient operations under both benign and adversarial conditions. While standardized communication protocols for eVTOL systems are still under development, the Micro Air Vehicle Link (MAVLink) protocol has emerged as a widely adopted solution for C2 communications in unmanned aerial systems (UAS). However, MAVLink lacks built-in, standardized encryption mechanisms, resulting in fragmented and inconsistent security implementations across platforms [5]. This inconsistency increases the risk of unintended behaviors and potential vulnerabilities.

The MAVLink protocol has been extensively used in UAV communications, but its application to eVTOL systems introduces unique security challenges. Amazon’s UAV delivery system, which shares some functions with eVTOL, highlighted the importance of signal authentication for command integrity. However, Amazon’s approach employed basic safety protocols, such as programming UAVs to return to a predefined location if communication was lost, and relied on staff to monitor multiple flights to ensure operational efficiency [6]. In contrast, eVTOL systems have significantly higher security requirements [7], [8].

Two main regulatory bodies play a role in the safety of C2 data exchange in UAVs; the FCC and the FAA. The Federal Communications Commission (FCC) is responsible for managing and licensing electromagnetic spectrum for commercial and public safety purposes, ensuring that communication systems, including those used in eVTOL operations, operate without harmful interference [9]. The FAA (Federal Aviation Administration) sets standards for aviation safety, including communication protocols that must be followed to ensure a secure and reliable transmission of C2 data [10]. Despite stringent FCC and FAA test requirements, C2 links remain vulnerable to malicious attacks [11].

To address this requirement, this paper purposed a LLM aided formal verification test bed to streamline protocol extension and formal verification. The protocol that will be subjected to the test bed is MAVLink 2. The MAVLink is an open and lightweight protocol utilized for real-time UAV telemetry and control. Due to its early integration with low-powered hardware in 2009, MAVLink retains a minimal overhead per packet, favoring reduced bandwidth and latency. As a result of this it has become widely adopted in academia and industry. MAVLink 1.0 protocol has zero security imple-

mentations, during the implementation period of that protocol, connectivity and battery power were of greater concern.

With efficiency as the target design goal, MAVLink 1 uses only 8 bytes of overhead per packet. MAVLink 2 has extended the protocol and has 14 bytes of overhead per packet, and 27 when using the optional signing feature [12]. MAVLink 2 Specifications has extended their time synchronization protocol to include `target_system` and `target_component` fields to increase the ability to detect TIMESYNC responses to authenticated requests.

Even with the MAVLink 2 extension of security features, MAVLink ensures message authenticity, but not encryption. This leaves the protocol vulnerable to various attacks including but not limited to; Replay Attacks, Denial of Service (DoS) [13]. Various encryption algorithms can enhance MAVLink security, including symmetric and asymmetric encryption. Low overhead algorithms like ES-CBC, AES-CTR, RC4, and ChaCha20 are suitable for UAVs due to their limited processing power and memory [14].

The contributions of this paper are summarized as follows:

- **LLM Aided Protocol Translation:** We propose a novel prompt-driven formal verification framework that LLMs to automate the translation of MAVLink protocol specifications into formal verification models (nuXmv and ProVerif). This reduces the need for manual intervention and accelerates the verification workflow.
- **State Machine Modeling and Symbolic Verification:** We construct a detailed communication state machine for MAVLink 2 and translate it into nuXmv compatible symbolic transition systems. These models are automatically lifted into ProVerif to analyze cryptographic security properties under the Dolev Yao adversarial model.
- **Simulation and Over the Air Evaluation:** We implement a three stage testbed comprising simulation based attack execution (via Damn Vulnerable Drone on Gazebo/Ardupilot), LLM integrated formal reasoning, and over the air (OTA) testing on physical drones.

II. SYSTEM DESIGN

The goal was to establish a rapid testbed to support verification and validation (V&V) for the commonly used eVTOL protocol MAVLink 2 and to further validate the PAVE framework. This paper does not focus on all possible vulnerabilities, but is used to implement this novel framework and test bed onto an adjacent protocol to further test its feasibility.

Our previous work on Prompt Aided Formal Verification (PAVE) [15] demonstrated that Large Language Models (LLMs) can be effectively integrated into formal verification workflows by automating the translation of natural language specifications into formal properties suitable for tools like nuXmv and ProVerif. Originally applied to the 5G RRC layer, PAVE showed how prompt-based reasoning can identify liveness and safety properties, which were validated both in virtual and over-the-air (OTA) environments. This prior

framework established a four-stage pipeline: (1) specification preparation, (2) prompt-based inspection, (3) symbolic verification, and (4) OTA validation. The methodology significantly reduced the human effort typically required to verify complex telecom standards and demonstrated reproducible results in both simulation layers and real-world testbeds.

The purposed framework is shown in Figure 1, the system is broken three distinct stages; Prompt Aided Inspector, Simulation, and Over-the-air (OTA) Platform. Each stage is sequentially fed into the next with Subject Matter Expert (SME) approval. This section further details the intricacies of each subsystem and the value they add.

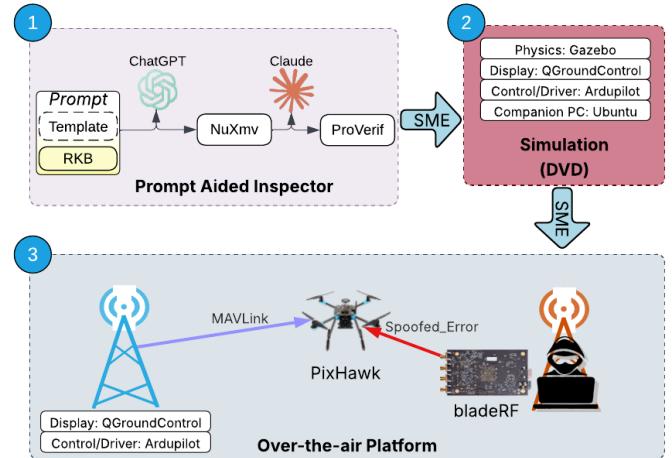


Fig. 1. Mobilized System Design of Prompt-Based Aided Formal Verification (PAVE) applied to MAVLink, facilitated with three connected components: 1. Prompt-Based Inspector (Formal Verification), 2. Simulation, and 3. Over-the-Air Validation. Prompt-Based Inspector replaces human conversion of specifications to formal properties.

A. Security Features and Limitations of MAVLink 2

As previously stated, MAVLink 2 has minimal security implemented. Figure 2 shows the format and content of a typical MAVLink 2 frame. The security related content of the message is the CHECKSUM and SIGNATURE. The CHECKSUM is primarily used to check for errors in the data transmission, but the SIGNATURE provides authenticity and integrity. The SIGNATURE is 13 bytes in size and is composed of a 1 byte linkID, 6 byte TIMESTAMP, and a 6 byte SIGNATURE. The 6 byte SIGNATURE generated by a combination of the complete packet, timestamp, and secret key. When MAVLINK_IFLAG_SIGNED is set, then MAVLink 2 augments each signed packet with an additional 13-byte signature block of $\text{SHA256}_{48}(\text{secret_key} \parallel \text{header} \parallel \text{payload} \parallel \text{CRC} \parallel \text{linkID} \parallel \text{timestamp})$

The lack of security measures applied to MAVLink 2 protocols allows for numerous attacks to occur. The below identifies attacks and categorizes based on the classic CIA triad. Previous works have explored the possible threats models applied to MAVLink 2 protocol, our purpose is to further exercise a rapid formal verification framework. [13], [16]

MAVLink 2 Frame (12-280)		
CONTENT	DESCRIPTION	SIZE (BYTES)
STX	Start	1
LEN	Payload Length	
INC FLAGS	Incompatibility Flags	
CMP FLAGS	Compatibility Flags	
SEQ	Packet Sequence #	
SYS ID	System ID (sender)	
COMP ID	Component ID (sender)	
MSG ID	Message ID / Type	3
PAYLOAD	Data of Value	0-255
CHECKSUM	Checksum	2
SIGNATURE	Signature (Optional)	13

Fig. 2. MAVLink 2 Frame showing the various content and size of data packets in the protocol. Security related features of the protocol are highlighted in green including the optional Signature field. The Payload contains data of value allowing commands to be sent

1) *Confidentiality*: Confidentiality prevents unauthorized viewing, without data link encryption messages are able to be viewed in clear text. Potential attacks include; eavesdropping, identity spoofing, hijacking, unauthorized access, interception.

2) *Integrity*: Guarantees accuracy, consistency, and trustworthiness of information and systems by protecting against unauthorized modification or corruption. Potential attacks include; packet injection, replay attack, message deletion, message modification.

3) *Availability*: Ensures that information and services are accessible to authorized users when requested. Potential attacks include; command and control, jamming, routing attack, denial of service, GCS Spoofing.

The attack focused on the working example in this paper utilizes packet injection to reach an error state in the drone. This effectively fools the drone into thinking it has a critical IMU error present and initiates a return sequence for the drone to land. This example violates all three categorizations of the CIA Triad.

B. Integrating PAVE for MAVLink 2 Security Verification

The proposed system leverages our previously developed Prompt Aided Formal Verification (PAVE) framework [15], originally designed for 5G protocol verification, and adapts it to systematically analyze MAVLink 2. PAVE automates the translation of protocol specifications into Symbolic Model Verifier (SMV) transition systems for NuXmv based model checking, and then lifts those models into ProVerif for symbolic (Dolev-Yao) verification. This integration enables both trace level safety property analysis and cryptographic security verification within a unified, prompt driven workflow.

To apply the PAVE framework for MAVLink 2 formal verification, we first defined a communication state machine (SM) that captures the essential operational phases of the MAVLink 2 protocol. As shown in Figure 4, the SM is constructed to facilitate model checking and vulnerability discovery. The state machine defines the following states:

Extractpolated Prompt Template	
Fields	Options
LLM Model	GPT-4.1, GPT 4o, o4-mini-high, o3, Claude Sonnet 4
Role	Security Researcher using Formal verification methods to transform communication protocols to nuXmv and ProVerif models
Domain	5G OR MAVLink 2
Roles / Agents	5G: UE, gNB, AMF OR MAVLink: GCS, UAV
Message Set	Full protocol, selected messages
SM States	Generate OR provided in reference
Attack Model	Dolev-Yao full control, replay only, drop/inject, no crypto break
Security Goals	Full CIA, Secrecy(K), Integrity (M)
Abstract Level	Full detail, simplified fields, timers as counters, omit optional messages
Assumptions	Channels, attack model bounds, timers, omissions
Reference Data	Protocol Definitions, nuXmv formats, Proverif Formats
Output Contract	Definitive expected outputs

Fig. 3. Prompting template utilized in the PAVE framework. The Role field remains constant while all other fields are displayed with their applicable options. Option descriptions are truncated for a concise extrapolated view.

- **INIT:** The system powers on and begins basic initialization.
- **WAIT_HEARTBEAT:** The link remains idle until a valid HEARTBEAT is received from a peer, signaling presence.
- **HANDSHAKE_START:** If message signing is enabled, an optional cryptographic handshake is initiated to verify authenticity.
- **SYNC:** The link confirms the peer’s system_id and component_id; identity is now synchronized.
- **MESSAGE RECEIVED:** A MAVLink packet is parsed, authenticated, and queued for handling.
- **READY:** Normal operation—messages can be exchanged continuously until a timeout or error occurs.
- **ERROR:** If a CRC mismatch, signature failure, or malformed packet is detected, communication resets and transitions to **WAIT_HEARTBEAT**.
- **TIMEOUT:** No valid traffic is received within the allowed interval; the link enters an idle state awaiting renewed activity.

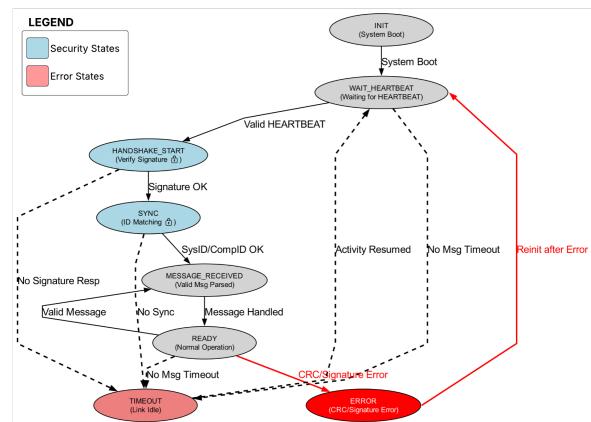


Fig. 4. MAVLink 2 communication state machine showing transitions from system initialization through heartbeat detection, optional handshake, synchronization, and message exchange.

After constructing the state machine, we generated a graphical representation in the common .DOT format, which was

then converted into .SMV format for symbolic verification using nuXmv. NuXmv is the industrial-strength evolution of NuSMV adding extensions such as; unbounded data types, arrays, clocks, real arithmetic, and AEAD-friendly bit-vectors. Computation Tree Logic (CTL) and Linear Temporal Logic (LTL) are leveraged to formally verify the MAVLink 2 state machine. CTL clauses expose reachability and invariant gaps across the branching state-space, while LTL clauses refine those findings into trace-level safety and progress guarantees. We applied standard Computation Tree Logic (CTL) and Linear Temporal Logic (LTL) formulas to evaluate the protocol's safety and liveness properties. In particular, CTL was used to assess reachability and invariants (e.g., $\text{EF } \varphi$, $\text{AG } \varphi$), while LTL formulas (e.g., $\text{G } \varphi$) were employed to verify persistent safety constraints across execution traces.

A prompt template can now be constructed, taking full use of the extended context window of OpenAI's o3, which offers a context window of 200,000 tokens. The Prompt Template includes; tone, purpose, example output, expected output, and finally the Reference Knowledge Base (RKB) definition. The RKB contains vital reference data including; MAVLink 2 protocols definitions, two nuXmv examples, nuXmv user manual, two ProVerif examples and the Proverif User Manual. With the Prompt Template and RFB established, we can then feed that information into a LLM of our choice. Various GPT models were compared from GPT 4o, o4-mini-high, o3, and GPT-4.1. For focus of this paper will be utilizing o3. OpenAI's o3 is a transformer-based model that uses a advanced chain-of-thought (CoT) prompting called simulated reasoning. This allows the model to pause it-self complete a multi-step internal prompt to reason about more complex tasking. This model provided the best syntactically correct responses and effectively leveraged the RKB for nuXmv simulation.

Our system implements a multi-LLM approach, using a different LLM between the nuXmv stage and the ProVerif stage of the formal verification. This was due to OpenAI's model's failing to produce syntactically correct formulations. For this stage of verification the system makes use of Anthropic Claude Sonnet 4 [17]. Claude Sonnet 4 is a LLM designed for efficient reasoning and code generation tasks. Claude demonstrates systematic reasoning behaviors in formal verifications tasks.

An iterative feedback loop is setup from the compiler to the LLM, with SME verification once a successful ProVerif protocol has been achieved. After a baseline ProVerif model is constructed, we can then add additional queries to test other plausible attacks. We first construct the appropriate CTL clauses and then expand to LTL clauses once attack traces are found. Once the formal verification stage identifies potential attacks, we then move to the Simulation Stage.

C. Simulated Attack via Software-in-the-Loop Framework

Once a potential attack is discovered and verified, the framework employs attacks against a fully software-driven

drone simulation using the Damn Vulnerable Drone (DVD) architecture [18], as illustrated in Figure 5. DVD is an open-source research testbed that containerizes drone system components on top of Kali Linux to enable controlled attack simulation. The platform includes an ArduPilot flight controller that interfaces with the Gazebo simulator via UDP and MAVProxy, a QGroundControl ground station that connects over TCP/IP using MAVLink, and a companion computer running Ubuntu for processing telemetry, camera streaming, and WiFi emulation. This setup provides a realistic and modular environment to evaluate the impact of protocol vulnerabilities under adversarial conditions.

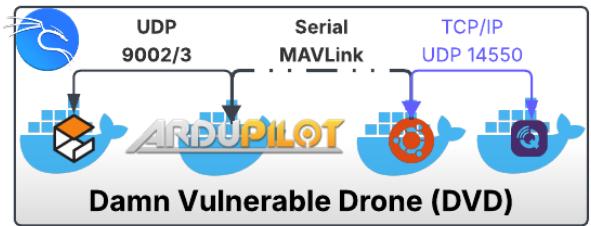


Fig. 5. DVD architecture with containerized components and MAVLink communication links running in a Kali Linux Virtual Machine.

D. Over-the-air Validation

The drone package utilized for this stage is the PX4 Development Kit - X500 V2 which is comprised of a Pixhawk 6X Autopilot Flight Controller, M8N GPS and SiK Telemetry Radio. The Radio Frequency (RF) Link utilized by the drone operates on both 433 and 915 MHz. We established a low altitude autopilot path on the campus of Stevens Institute of Technology in Hoboken, NJ. Once an automated flight path with emergency locations was programmed we then sought to establish a method to inject attacks on the MAVLink communication line.

A malicious GCS was established running on an Ubuntu Node outputting to a BladeRF device to amplify the signal over the air. First malicious MAVLink messages are generated in Python utilizing the PyMAVLink library. The raw MAVLink serial stream is modulated for SiK emulation using GNURadio. Gaussian Frequency Shift Keying (GFSK) is used with the following Nyquist calculated parameters: baud rate of 57600, symbol rate of 64,000/sec, frequency deviation of 25 kHz, sample rate of 512,000 and band width of 100 kHz. These messages are then transmitted via BladeRF at alternating center frequencies of 433 and 915 MHz.

III. IMPLEMENTATION AND RESULTS INTERPRETATION

This section will discuss the Critical Error Injection working example and trace it through the system. This attack has been previous identified [11], the novelty lies in the ability to rapidly prototype protocol extensions to solve known vulnerable states and potentially discover new ones. For a realistic way to test if these states become unreachable, the implementation of the stream cipher algorithm ChaCha20 was

chosen. ChaCha20 introduces a small performance overhead with CPU Processing and Memory Consumption within 2% of baseline MAVLink 2 without encryption. The total number of packets sent is within 5% of baseline. [16] Some works have been able to implement portions of ChaCha20 in a novel way to produce virtually no additional overhead. [19] [20]

A. nuXmv Symbolic Model Results

This section details the application of temporal logic formulas for evaluating the MAVLink 2 protocol's resilience to the injection of malicious messages. Table I summarizes the logic types and formulas used for each property under investigation. Table II compares the baseline MAVLink 2 implementation with the improved output after integrating ChaCha20 encryption.

1) *Critical_Error_Reachable* (CTL: $EF \varphi$): To evaluate the protocol's resilience to message injection, we specify the CTL formula $EF \varphi$, where φ denotes a critical error state. Satisfaction of this formula indicates that there exists at least one execution path on which a critical system error is reachable. This trace demonstrates that, under certain adversarial conditions, the MAVLink 2 protocol permits a sequence of events leading to a critical error, revealing a potential vulnerability to injected or malformed message events.

2) *Inject_After_Ready_Error* (CTL: $AG \varphi$): We use the CTL formula $AG \varphi$ to verify that, once the protocol transitions to the ready state, injected messages cannot trigger an error condition. Violation of this property indicates that message injection is possible even after successful initialization, highlighting the need for persistent authentication or validation throughout system readiness.

3) *Ready_Requires_TimeSync* (CTL: $AG \varphi$): The CTL clause $AG \varphi$ is used to confirm that transition into the ready state is always conditioned on the successful completion of a time synchronization event. This property enforces synchronization as a prerequisite for system readiness, ensuring temporal alignment prior to mission-critical operations.

4) *Drone_Reboots_After_Error* (CTL: $AG \varphi$): To validate protocol recovery, we assert the CTL property $AG \varphi$, requiring that detection of any protocol error is always followed by a reboot state. Satisfaction of this formula demonstrates that error-handling and recovery procedures are reliably invoked, reducing the risk of persistent error conditions.

5) *Safety_Violated_By_Injection* (LTL: $G \varphi$): We apply the LTL property $G \varphi$ to examine whether safety guarantees are preserved under repeated message injection. If this property is violated, it reveals that persistent adversarial inputs can compromise system safety, indicating the protocol's susceptibility to injection or denial-of-service style attacks.

6) *Inject_Event_Eventually_Error* (LTL: $G \varphi$): Using the LTL property $G \varphi$, we evaluate whether continuous message injection inevitably leads to an error state. Violation of this property shows that, under sustained injection conditions, the protocol accumulates errors, eventually reaching a

failure condition. This outcome highlights a lack of robustness against sustained adversarial input.

7) *TimeSync_Eventually_Success* (LTL: $G \varphi$):

The LTL property $G \varphi$ is specified to confirm that, across all execution traces, the occurrence of a time synchronization event eventually results in successful synchronization. This property verifies the protocol's ability to guarantee synchronization completion despite adversarial interference or network delay.

8) *Offset_Bounded_After_Sync* (LTL: $G \varphi$):

We define the LTL property $G \varphi$ to ensure that, following a successful synchronization event, the system's time offset remains within specified bounds for all future states. Satisfaction of this property indicates that the protocol maintains bounded temporal deviation after synchronization, validating post-sync stability under adverse or uncertain conditions.

TABLE I
NUXMV PROPERTIES FOR CRITICAL ERROR INJECTION VERIFICATION

#	Property Name	Logic	Meaning
1	Critical_Error_Reachable	$EF \varphi$	on some path
2	Inject_After_Ready_Error	$AG \varphi$	on all paths
3	Ready_Requires_TimeSync	$AG \varphi$	on all paths
4	Drone_Reboots_After_Error	$AG \varphi$	on all paths
5	Safety_Violated_By_Injection	$G \varphi$	globally
6	Inject_Event_Eventually_Error	$G \varphi$	globally
7	TimeSync_Eventually_Success	$G \varphi$	globally
8	Offset_Bounded_After_Sync	$G \varphi$	globally

TABLE II
BASELINE VS. CHACHA20 NUXMV VERIFICATION RESULTS FOR SPOOFED PACKET INJECTION.

#	Property Interpretation	Baseline	ChaCha20
1	Injected CRITICAL_ERROR can reach S_ERROR.	TRUE	FALSE
2	From any READY, injection still reaches S_ERROR.	TRUE	FALSE
3	READY not entered until TIMESYNC succeeds.	TRUE	TRUE
4	Injected error eventually leads to S_INIT restart.	TRUE	TRUE
5	Injected packet never coincides with S_ERROR.	FALSE	TRUE
6	Every inject eventually causes S_ERROR.	FALSE	FALSE
7	If TIMESYNC isn't blocked, it eventually succeeds.	FALSE	FALSE
8	After sync, offset stays within ±500 ms.	TRUE	TRUE

B. ProVerif Model Results

Following the construction and verification of the state machine in nuXmv, the protocol was translated to a symbolic model for analysis in ProVerif. The conversion process utilized Claude Sonnet 4 to automate the extraction of MAVLink 2 message formats and protocol states. The resulting ProVerif model incorporates explicit message constructors for heartbeat, command, and status messages, each parameterized with system identifiers, sequence numbers, payloads, and timestamp values in accordance with the MAVLink 2 specification.

For the security analysis, key protocol events were defined, including `DroneArmed()`, `DroneDisarmed()`, `CommandReceived(system_id, payload)`, and `MaliciousCommand(payload)`. The primary security queries focused on the reachability of the `DroneArmed()` state and the potential for an adversary to trigger the `MaliciousCommand(land_cmd)` event.

A multi-model approach was adopted to ensure robust validation, with Claude Sonnet 4 systematically generating and lifting the state machine constructs into ProVerif's input language. The final model enabled automated verification of the defined queries, and the analysis results, as summarized in Figure 6, indicate that both `DroneArmed()` and `MaliciousCommand(land_cmd)` are reachable under adversarial conditions, revealing exploitable vectors for command injection in MAVLink 2.

To address these vulnerabilities, the protocol model was extended to incorporate symmetric encryption of messages using the ChaCha20 cipher. In the modified ProVerif model extended with ChaCha20, the following changes were introduced:

- A symmetric key was defined and distributed to both communicating parties prior to protocol execution.
- All message constructors were updated to encapsulate the MAVLink 2 messages within a ChaCha20 encryption function, ensuring confidentiality and integrity of transmitted data.
- The protocol logic was adjusted so that message reception and processing require successful decryption using the shared key.

With these modifications, the ProVerif analysis was repeated using the same event definitions and security queries. The updated model verified that neither the `DroneArmed()` event nor the `MaliciousCommand(land_cmd)` event is reachable by the adversary under the assumed threat model. This result demonstrates that the introduction of ChaCha20-based symmetric encryption effectively mitigates the previously identified command injection attacks, providing strong security guarantees for MAVLink 2 communications. The verification outcome for the protected protocol is summarized in Figure 7.

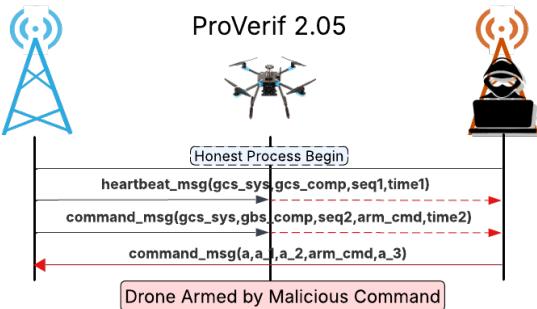


Fig. 6. ProVerif Traces of baseline MAVLink 2 Protocol. Showing an extrapolated ProVerif trace output focusing on the attacker being able to inject malicious commands.

C. Damn Vulnerable Drone (DVD) Evaluation

To further assess protocol vulnerabilities in operational contexts, the Damn Vulnerable Drone (DVD) framework was deployed. DVD is an open-source research testbed that enables controlled evaluation of UAV security by supporting message interception, injection, and modification between the

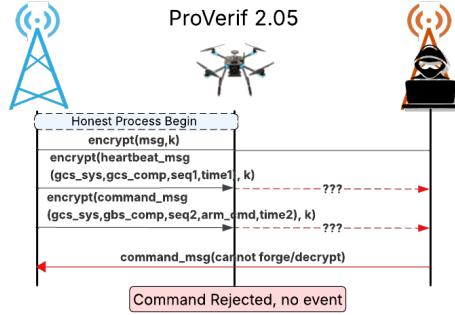


Fig. 7. ProVerif Traces of MAVLink 2 Protocol with ChaCha20 extension. The adversary is no longer able to decrypt messages or forge new ones. The process ends with no malicious events being triggered.

ground control station, UAV autopilot, and adversarial relay, as illustrated in Figure 5.

The DVD environment was used to validate formal findings under practical conditions. In the baseline configuration, injection of unauthorized messages resulted in observable unsafe drone behavior, consistent with the symbolic model's predicted attack traces. Notably, targeted adversarial packets triggered a critical IMU (Inertial Measurement Unit) error, causing the UAV to report loss of IMU data synchronization and engage failsafe procedures such as forced landing or mode transition. This confirms that unprotected channels are susceptible to denial-of-service and sensor disruption attacks.

Upon integration of ChaCha20-based encryption into the protocol stack, all adversarial injection attempts were rejected at the interface, with no effect on UAV state or IMU subsystem integrity. These results underscore the importance of robust cryptographic mechanisms for protecting UAV control channels against protocol-layer exploits.

D. Over-the-Air (OTA) Validation

The protocol was further evaluated in a live over-the-air setting using the PX4 Development Kit – X500 V2. Testing was conducted at Stevens Institute of Technology, where the drone executed a preprogrammed autonomous flight. A dedicated adversarial ground control station (GCS), equipped with software defined radio, was used to inject malicious MAVLink commands into the control link during flight.

In the baseline configuration, the drone accepted injected critical error messages, resulting in immediate transition to an emergency hover state. This outcome directly validates the findings of the symbolic and simulated analyses, demonstrating that message injection vulnerabilities are exploitable in operational environments.

Subsequent trials incorporating cryptographic protection on the MAVLink channel prevented successful injection; adversarial messages were ignored, and the drone maintained nominal operation throughout all attack attempts. These results highlight the necessity of protocol-layer security for UAV operations in contested environments.

IV. DISCUSSION

The results of this study highlight the practical challenges and considerations involved in leveraging LLMs for protocol verification and symbolic analysis. We adopted a multi-model workflow.

OpenAI's o3 model demonstrated strong performance for nuXmv-compatible state machine synthesis and temporal logic property extraction, producing verifiable models with minimal iteration. However, attempts to use OpenAI's models for ProVerif code generation consistently resulted in invalid syntax or incomplete constructs.

In contrast, Claude Sonnet 4 exhibited improved capability for symbolic protocol modeling with ProVerif. To overcome syntax errors in zero-shot prompting, we employed a multi-shot design. By incorporating user-generated examples and iteratively refining outputs through feedback, Claude was able to produce valid ProVerif models that handled complex cryptographic primitives. This multi-LLM approach ultimately proved necessary to meet the specification and verification demands of both nuXmv and ProVerif within our workflow.

These findings suggest that, despite ongoing advancements, LLMs remain specialized in their effective domains. Careful model selection and prompt engineering remain essential for integrating LLMs into formal verification pipelines for secure systems design.

V. CONCLUSION

This study demonstrates the feasibility and effectiveness of a multi-stage, LLM-assisted formal verification platform for UAV communication protocols. By combining automated prompt driven translation with nuXmv and ProVerif, the workflow enables end-to-end vulnerability discovery, simulation, and OTA validation for MAVLink 2. Results from both symbolic analysis and live testing confirm that baseline MAVLink 2 is vulnerable to injected command attacks. These injected commands are capable of disrupting flight operations and critical subsystems. Integrating ChaCha20 based symmetric encryption into the protocol model eliminates these attack vectors, with security properties validated across all verification stages. The multi LLM strategy, leveraging OpenAI for state machine modeling and Anthropic Claude for cryptographic protocol analysis, proved essential for complete and correct model synthesis. This work supports the adoption of LLM accelerated formal verification and robust cryptographic extensions as critical enablers for secure, resilient UAV systems.

VI. ACKNOWLEDGMENT

This Research was sponsored by the Army Research Office and was accomplished under Grant Number W911NF-25-1-0183. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

REFERENCES

- [1] J. Meharg, T. Byrnes, J. Sabatino, S. Paradkar, E. Crabtree, A. Khan-delwal, and Y. Wang, "Enhanced uav classification: Gaining deeper insights from mechanical vibration over rf characteristics," in *2025 IEEE Aerospace Conference*. IEEE, 2025, pp. 1–9.
- [2] J. Yang and Y. Wang, "Toward auto-modeling of formal verification for nextg protocols: A multimodal cross-and self-attention large language model approach," *IEEE Access*, vol. 12, pp. 27 858–27 869, 2024.
- [3] J. Yang, S. Arya, and Y. Wang, "Formal-guided fuzz testing: Targeting security assurance from specification to implementation for 5g and beyond," *IEEE Access*, 2024.
- [4] J. Yang and Y. Wang, "Trustworthy formal verification in 5g protocols: Evaluating generative and classification models for relation extraction," in *MILCOM 2024 - 2024 IEEE Military Communications Conference (MILCOM)*, 2024, pp. 956–962.
- [5] I. Hughes, A. Pupo, J. Wynd, Z. Thurlow, C. Ivancik, and Y. Wang, "Securing the unprotected: Enhancing heartbeat messaging for mavlink uav communications," in *2024 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*. IEEE, 2024, pp. 1–6.
- [6] G. Brunner, B. Szebedy, S. Tanner, and R. Wattenhofer, "The urban last mile problem: Autonomous drone delivery to your balcony," in *International Conference on Unmanned Aircraft Systems (icuas)*, 2019, pp. 1005–1012.
- [7] M. Gharibi, R. Boutaba, and S. L. Waslander, "Internet of drones," *IEEE Access*, vol. 4, pp. 1148–1162, 2016.
- [8] B. Mak, M. Mansouri, and Y. Wang, "Introduction of electric vertical takeoff and landing system with systemigram approach," in *2023 18th Annual System of Systems Engineering Conference (SoSe)*. IEEE, 2023, pp. 1–8.
- [9] Federal Communications Commission, "Fcc rules and regulations," 2021.
- [10] Federal Aviation Administration, "Faa regulations and policies," 2020.
- [11] Y.-M. Kwon, J. Yu, B.-M. Cho, Y. Eun, and K.-J. Park, "Empirical analysis of mavlink protocol vulnerability for attacking unmanned aerial vehicles," *IEEE Access*, vol. 6, pp. 43 203–43 212, 2018.
- [12] S. Atoev, K.-R. Kwon, S.-H. Lee, and K.-S. Moon, "Data analysis of the mavlink communication protocol," in *2017 International Conference on Information Science and Communications Technologies (ICISCT)*. IEEE, 2017, pp. 1–3.
- [13] M. Ficco, R. Palmiero, M. Rak, and D. Granata, "Mavlink protocol for unmanned aerial vehicle: Vulnerabilities analysis," in *2022 IEEE Int'l Conf on Dependable, Autonomic and Secure Computing*. IEEE, 2022, pp. 1–6.
- [14] V. Satheesh and D. Shanmugam, "Implementation vulnerability analysis: A case study on chacha of sphincs," in *2020 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS)*, 2020, pp. 97–102.
- [15] T. Wray and Y. Wang, "5g specifications formal verification with over-the-air validation: Prompting is all you need," in *MILCOM 2024-2024 IEEE Military Communications Conference (MILCOM)*. IEEE, 2024, pp. 412–418.
- [16] N. Sabuwala and R. D. Daruwala, "Securing unmanned aerial vehicles by encrypting mavlink protocol," in *2022 IEEE Bombay Section Signature Conference (IBSSC)*, 2022, pp. 1–6.
- [17] "Claude System Card: Claude Opus 4 and Claude Sonnet 4," Anthropic, Tech. Rep., May 2025. [Online]. Available: <https://www-cdn.anthropic.com/6be99a52cb68eb70eb9572b4cafad13df32ed995.pdf>
- [18] N. Aleks, "Damn vulnerable drone simulation platform," <https://github.com/nicholasaleks/Damn-Vulnerable-Drone>, 2022.
- [19] N. A. Khan, N. Jhanjhi, S. N. Brohi, A. A. Almazroi, and A. A. Almazroi, "A secure communication protocol for unmanned aerial vehicles," *CMC-Computers Materials & Continua*, vol. 70, no. 1, pp. 601–618, 2022.
- [20] A. Allouch, O. Cheikhrouhou, A. Koubâa, M. Khalgui, and T. Abbes, "Mavsec: Securing the mavlink protocol for ardupilot/px4 unmanned aerial systems," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 2019, pp. 621–628.