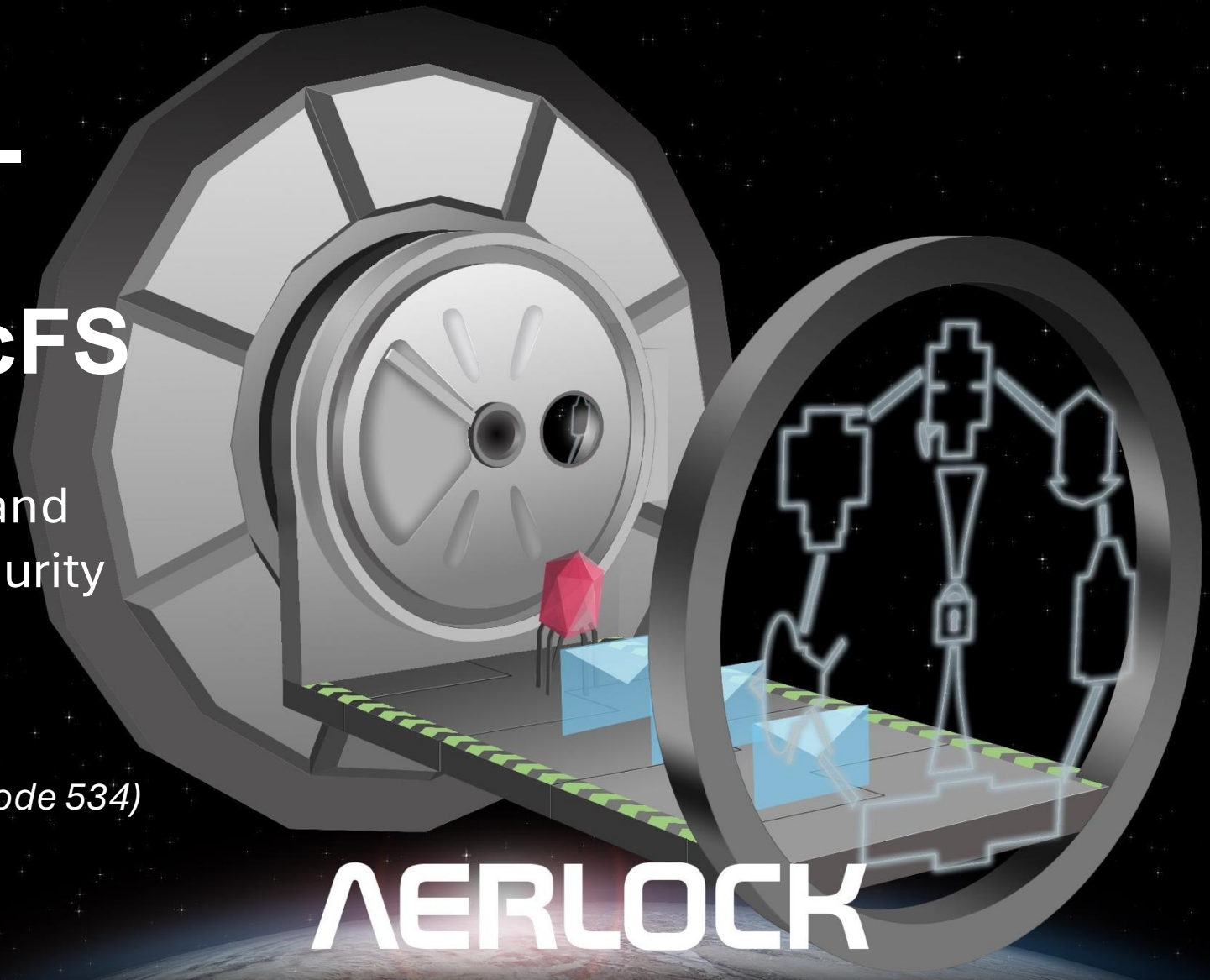
AerLock: Flight-Grade Secure Messaging for cFS

Authenticated, Observable, and Operationally Controlled Security

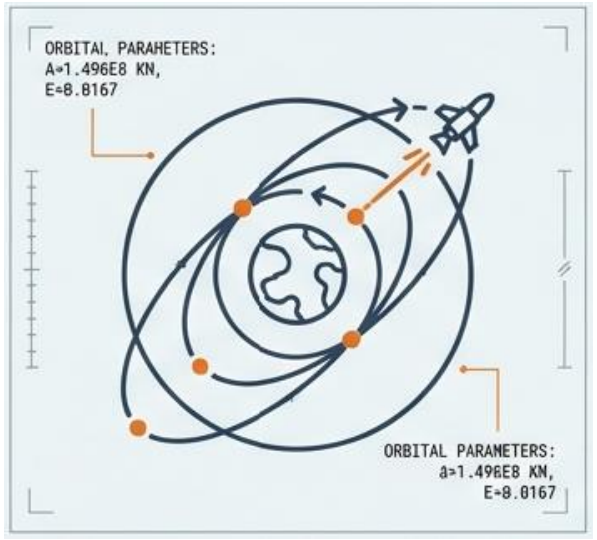
Eshwar Singh, Eric Pollack, &
Alex Schoening (*Vantage Systems*)

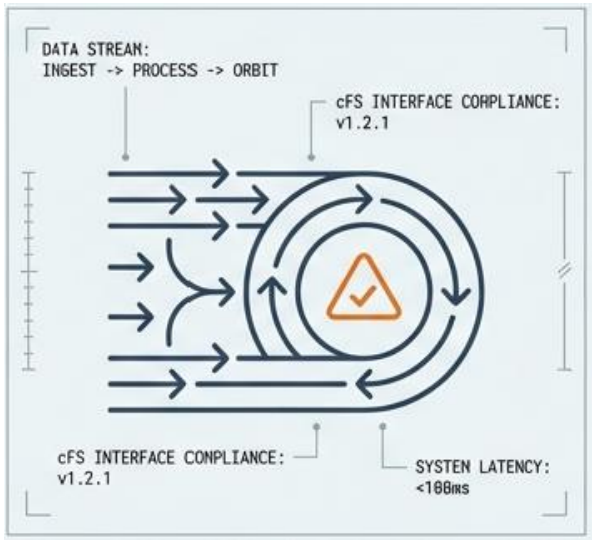
NASA Goddard Space Flight Center (Code 534)

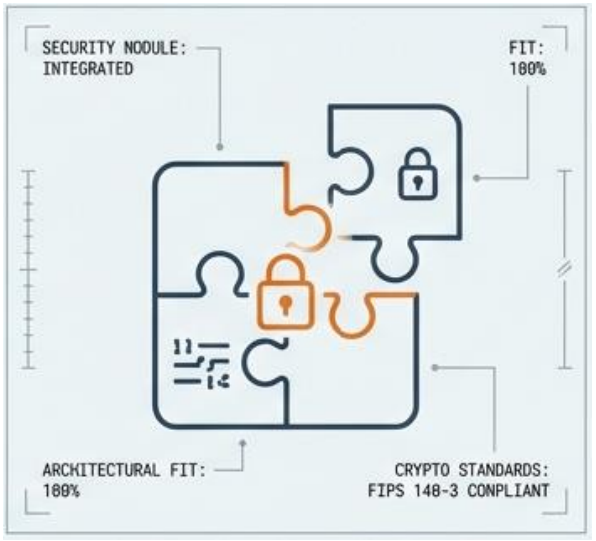


Built on Operational Reality

Developed by a multi-disciplinary team spanning flight software, cybersecurity, and mission operations.







Mission-Driven

Designed in collaboration with mission partners and integrator to reflect real spacecraft constraints.

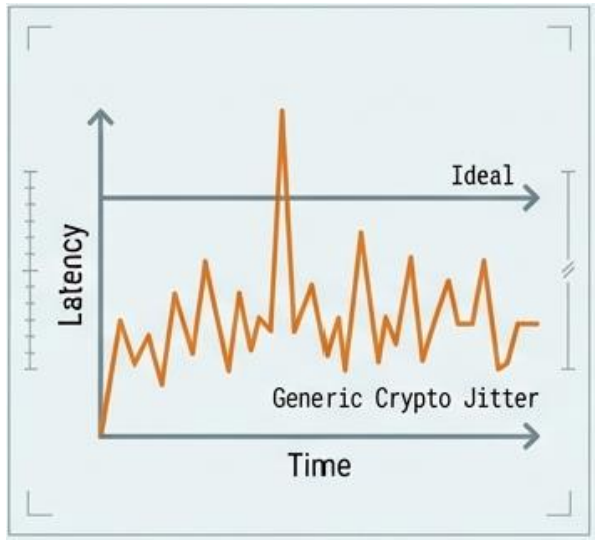
Workflow Compatible

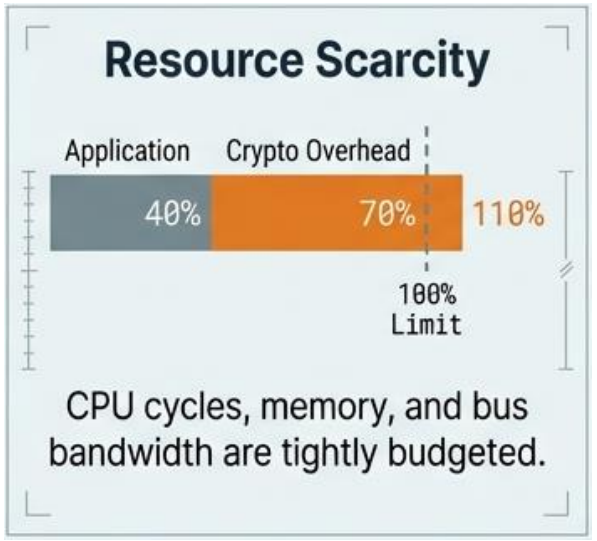
Built to integrate cleanly with existing cFS missions; it does not require re-architecting the system.

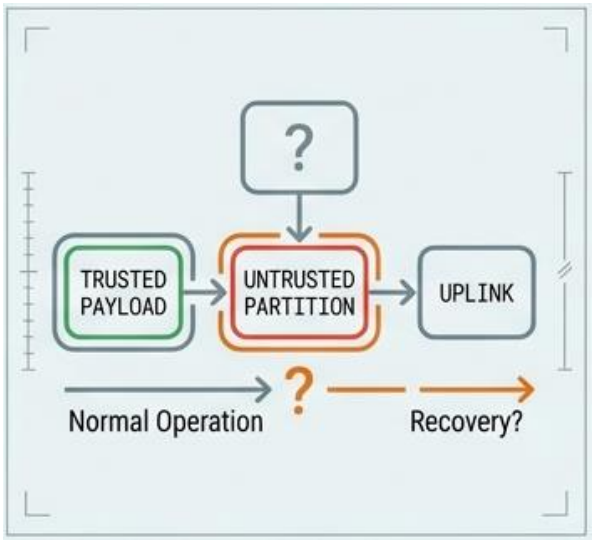
Clean Integration

Not “bolted on” afterwards – security is designed as part of the cFS flight architecture.

Why ‘Just Add Crypto’ Fails in Flight Software







Real-Time Determinism

Flight software requires bounded latency. Security must not introduce jitter, blocking, or unbounded execution time.

Mixed-Trust Environments

Systems are not monolithic; they contain payloads, partitions, and uplink paths with varying trust levels.

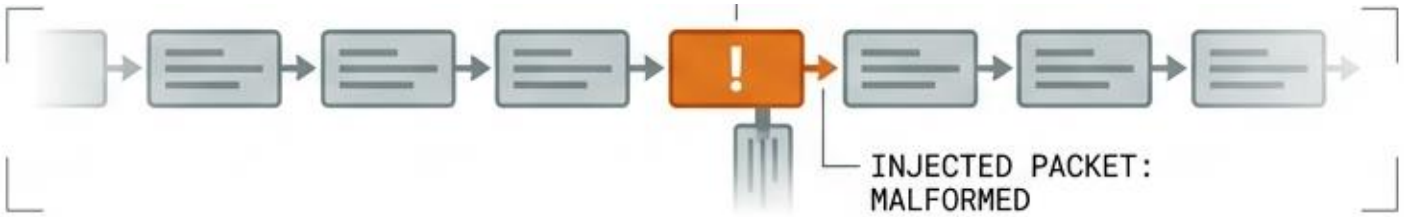
Operational Conditions

Must handle reboots, intermittent links, and partial failures gracefully.

The Threat Landscape

Message Injection

Unauthorized or malformed messages triggering unintended spacecraft behavior.



Tamper & Corruption

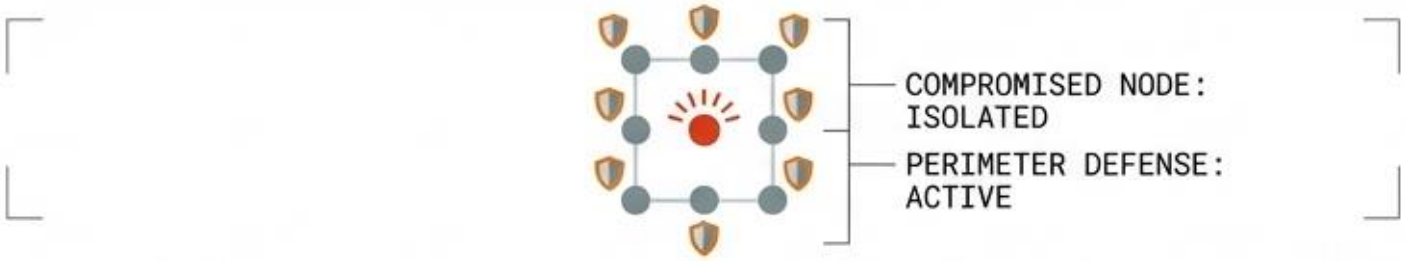
The necessity to distinguish hardware faults from adversarial modification.

Replay Attacks

Replaying valid messages out of context (dangerous for time-shifted missions).

Blast Radius

A Compromised key must not expose the entire system or constellation.



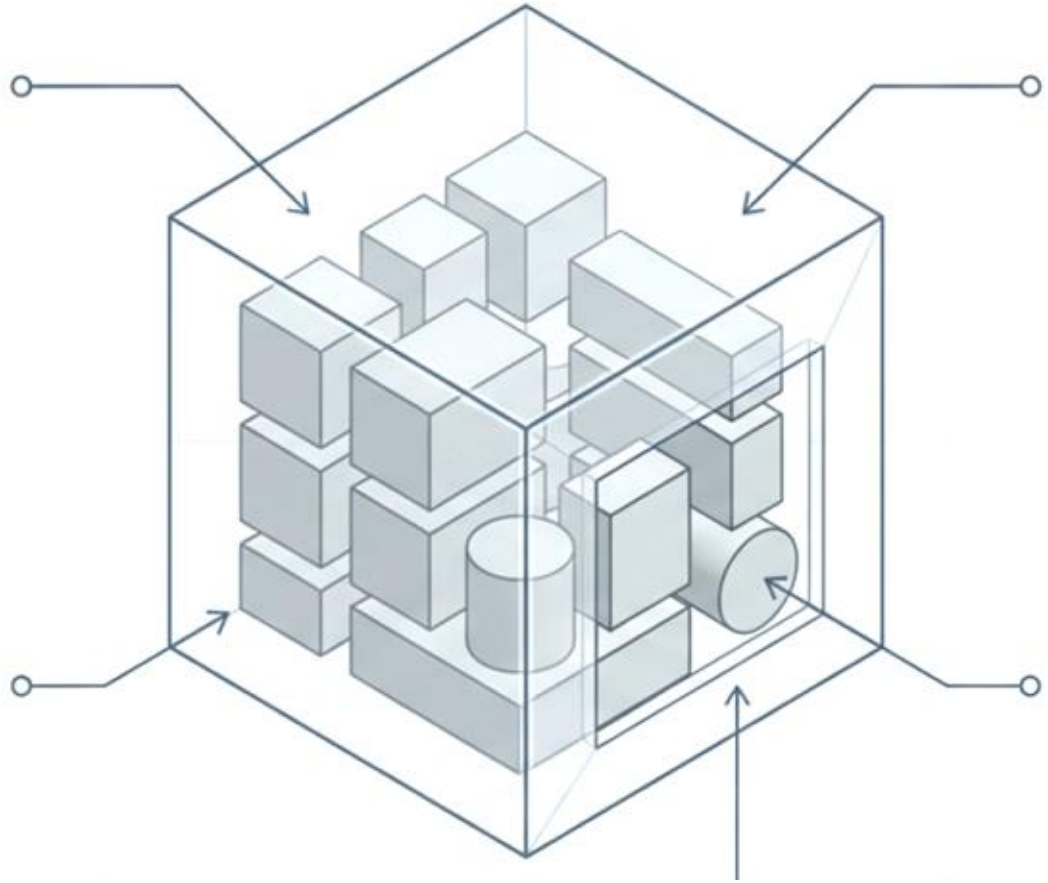
Design Goals for Embedded Security

Minimal Friction

Small API Surface, inline usage, and compatibility with existing mission apps.

Predictable Performance

Bounded CPU and memory usage. No dynamic allocation or hidden costs



cFS-Native

Preserves standard concepts; pipes, message IDs, tables, and events.

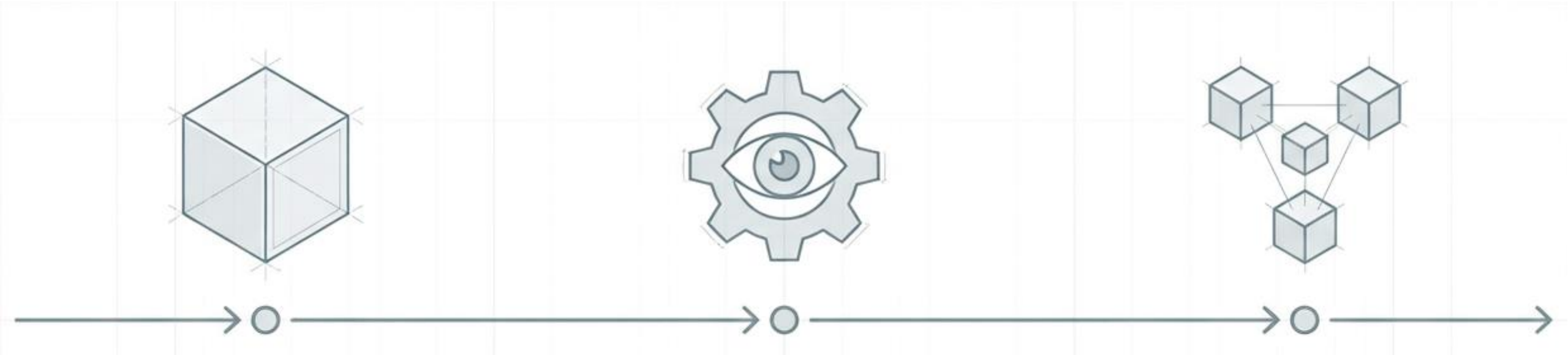
Crypto Agility

Algorithms/suites can evolve without application rewrites

Visibility

Security behavior is observable through telemetry – not a black box

The Evolution of Trust



AerLock v1

- Intra-node protection
- Authenticated messaging within a single cFS instance.
- Focus on protecting on-board commanding and enforcing trust boundaries.

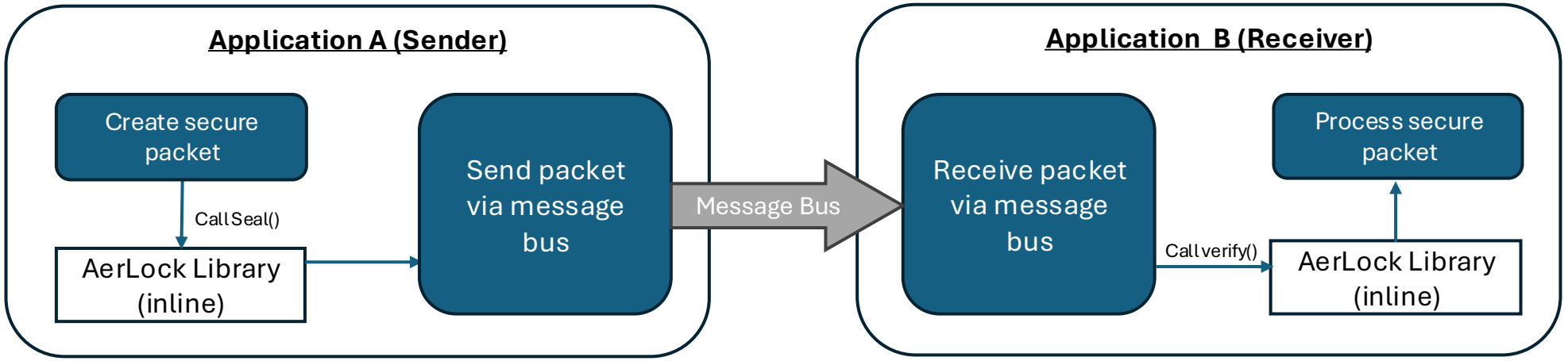
Security Manager

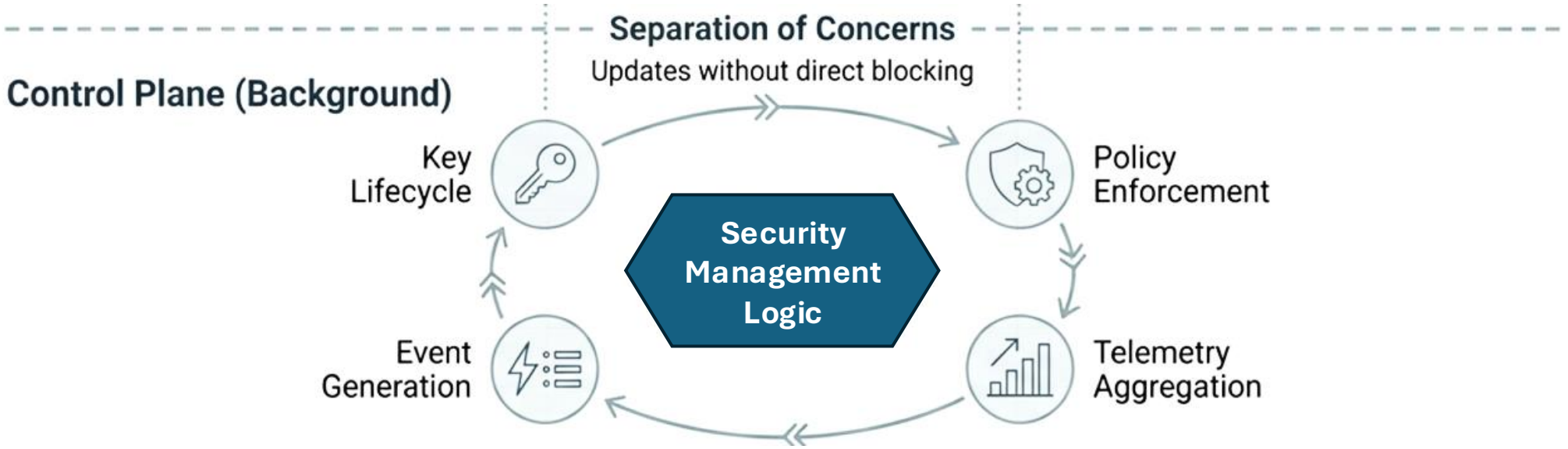
- Centralized policy, provisioning, and operational control
- Security Association (SA) orchestration
- Key generation, activation, & rotation

AerLock v2

- Inter-node constellation
- Secure, authenticated communication across multi-node constellations, aligned with SDLS standards

AerLock v1: Architecture





A Low-Friction Integration Model

1. Link & Include

Add the AerLock library and headers. No special runtime dependencies required

2. Register

Establish identity and configuration during application startup

```
AerLock_Register(AppID, Config);
```

3. Seal & Verify

Seal messages on transmit. Verify on receive. Run the background manager.

```
AerLock_Seal(MsgPtr);
...
AerLock_Verify(MsgPtr);
```

The Role of the Security Manager

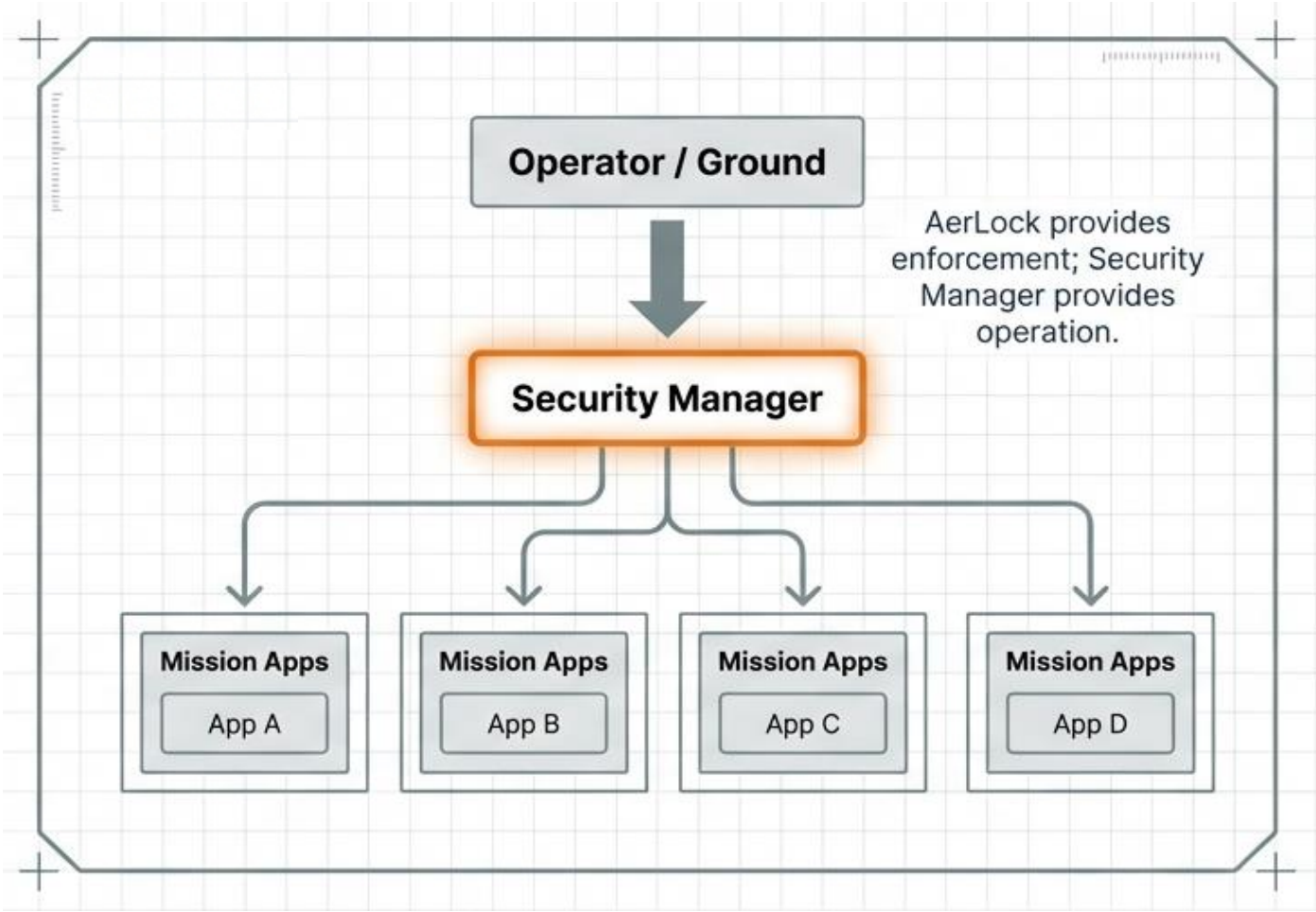
Decoupling Policy from Code

Why it is Required:

- **Policy Decoupling**: Applications should not hardcode security decisions.
- **Centralized Governance**: One place to manage mission-wide security posture.

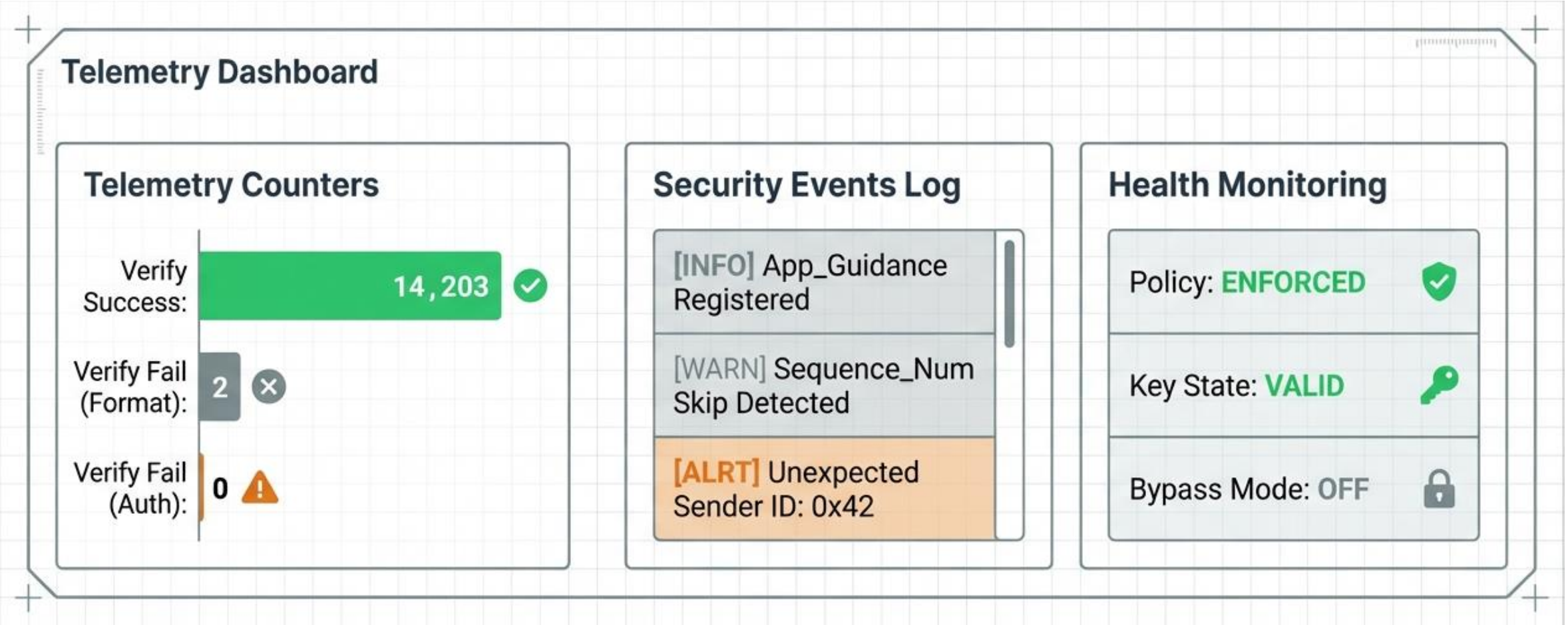
Responsibilities:

- Central policy authority
- Provisioning and key management coordination
- Aggregation of security telemetry
- Operator command and control interface



Runtime Visibility & Observability

Distinguishing hardware faults from attacks.



Configuration & Response Actions

Flexibility without recompilation

Configuration Matrix

	App/Interface	Posture	Crypto Suite	Failure Action
1	Guidance & Nav	ENFORCED	AES-GCM-256	DROP
2	Science Payload	DEGRADED	ChaCha20-Poly1305	LOG_ONLY
3	Debug Stream	BYPASSED	None	IGNORE

Operational Response Actions:

- Drop
- Alert
- Log
- Degrade

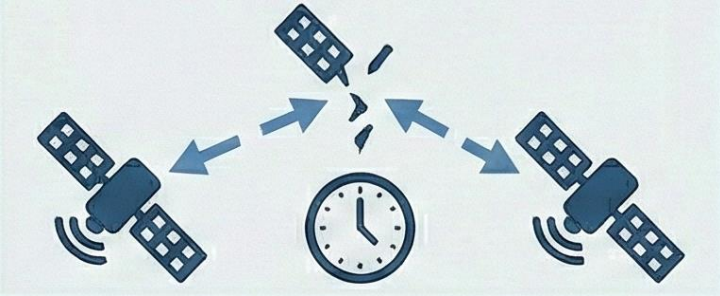
Mechanism: All changes take effect through standard cFS tables and commands

Constellations Need a Different Security Model

Standard SDLS assumptions do not hold in store-and-forward environments. AerLock v2 bridges the gap.

The Constellation Problem Space

- Standard SDLS implementations assume continuous links and synchronized peers.
- Constellations operate with:
  - Intermittent connectivity & store-and-forward delays
- Asymmetric recovery scenarios.
- Partial trust between nodes.

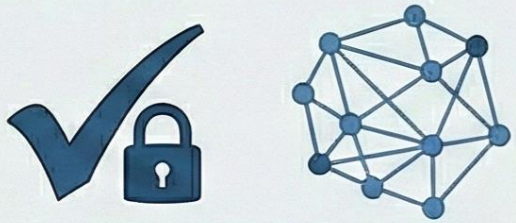


What AerLock v2 Changes

- Adapts SDLS for reality by shifting the model
- **Trust Scope**: Relationship-scoped, not global
- **State Management**: Persistent, not ephemeral.
- **Traffic Handling**: Valid messages remain verifiable across significant delays.
- **Recovery:** Explicit and stateful, not implicit or timing-based

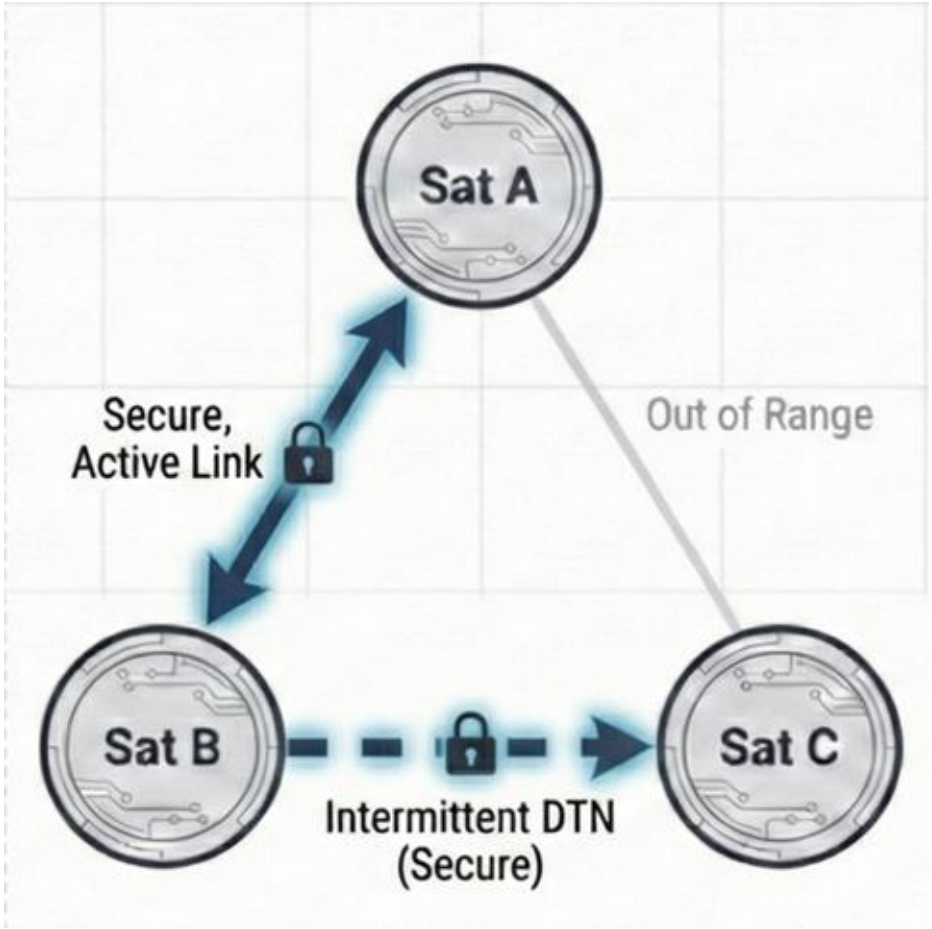
The Outcome

- Robust security that survives operations:
- Secure cross-node messaging without breaking SDLS compliance
- Prevents global rekeying storms after outages
- Eliminates “silent” trust resets after outages.
- Removes assumption that “everyone is online now”



Designed for constellation and store-and-forward environments

AerLock v2: Constellation Security

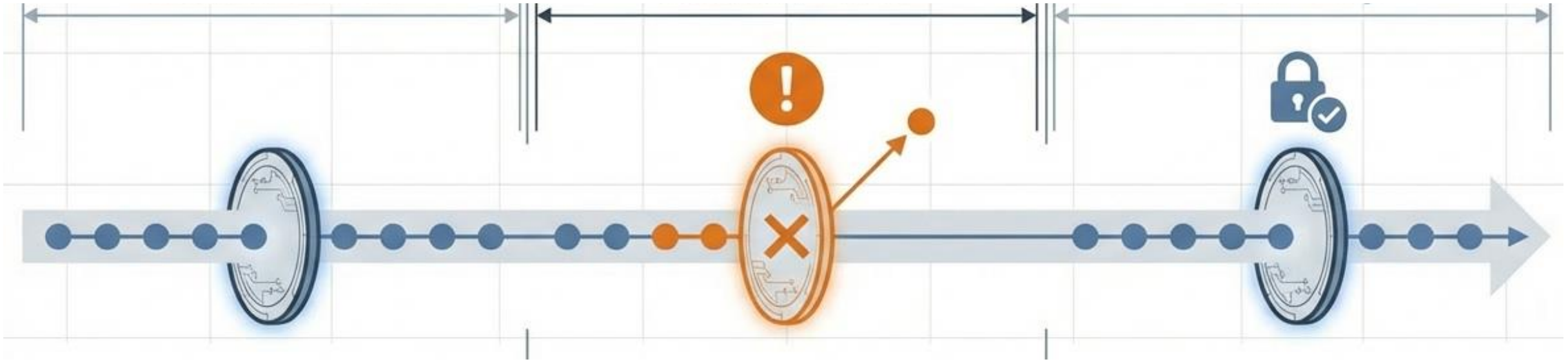


Core Features

1. **CCSDS SDLS Alignment**
   Compliant with 355.0 / 355.1 Data-plane protection
2. **Relationship-Bound Trust**
   Security contexts bound to specific sender-receiver pairs
3. **Delay-Tolerant Operation**
   Valid messages accepted across delayed delivery windows
4. **Outage Recovery**
   - **Dormant**: Security state preserved offline
   - **Recovering**: Rejects stale messages while re-establishing trust.

Operational Scenario Walkthrough

Normal Traffic | Attack / Fault | Recovery



Seal/Verify Operations Running

Healthy Telemetry

Verification Failure

Security event generated. Message safely dropped.

Trust Re-established

Traffic resumes with operator visibility

Summary

Key Takeaways:

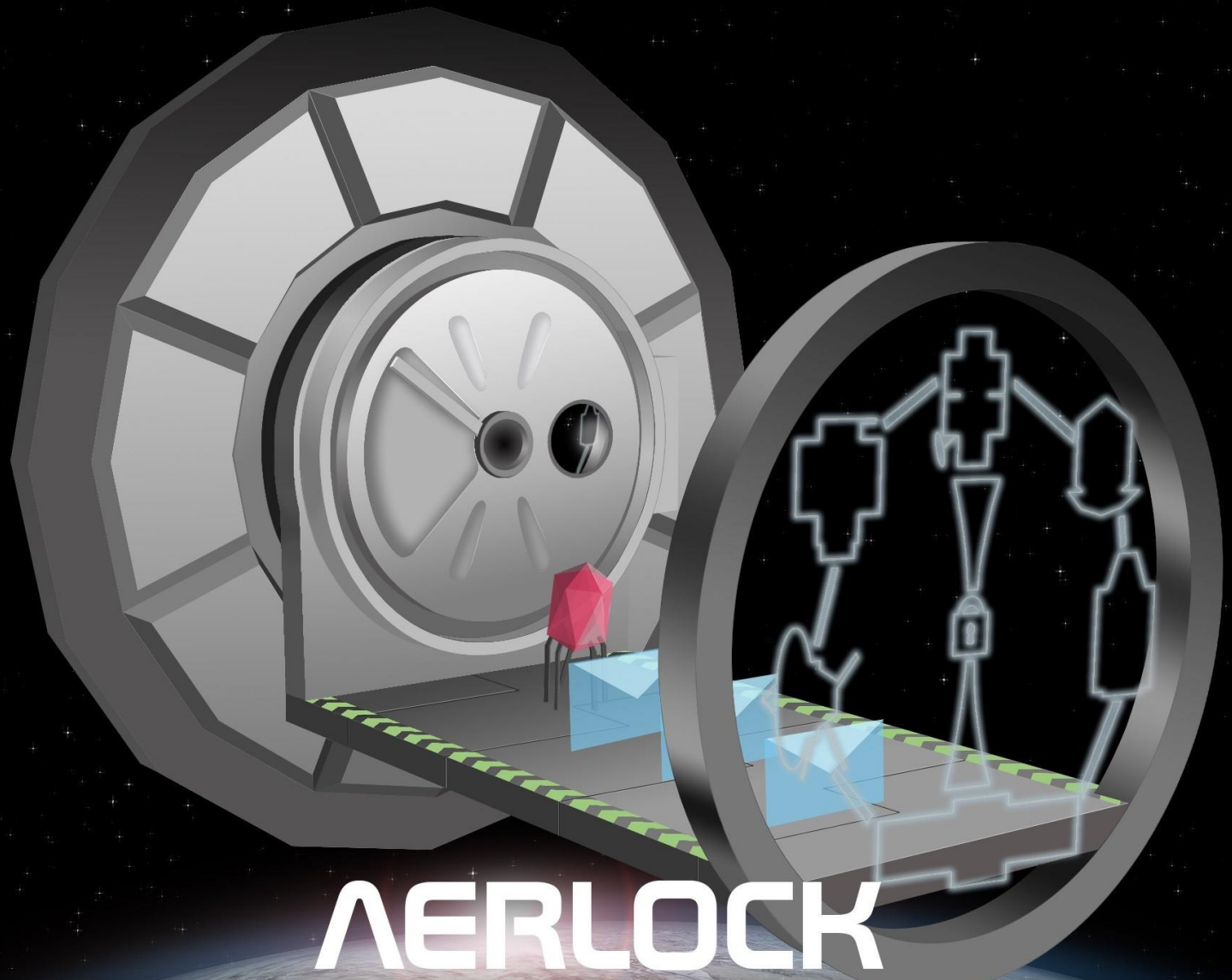
1. **AerLock v1**: Enables authenticated messaging that fits cFS workflows today.
2. **Security Manager**: Provides the policy and operational control required for flight
3. **AerLock v2**: Extends trust across links, nodes, and time for constellation missions

For More Information

For a demo or follow-on questions about how the AerLock system can secure your mission:

Contact:
Dr. Ashok Prajapati
ashok.k.prajapati@nasa.gov
NASA GSFC - Code 534

Questions?



Backup

Embedded Security for cFS Missions

- Protecting on-board commanding.
- Authenticating Telemetry.
- Enforcing Trust.

AerLock		Security Manager
v1 (Application Layer) – Data Plane Intra-cFS Application Messaging	**v2 (Link Layer) – Data Plane** Inter-node/ Cross-link Messaging	**Control Plane** Security authority + Orchestration
• CCSDS Space Packet (133.0-B-2) • Message authentication & encryption • Anti-replay Protection (app-Level scoped) • Dynamic session key management • In-line, data path protection • Platform-agnostic (RTEMS, Linux,..)	• CCSDS SDLS (355.0/ 355.1) • Secure transfer frame protection (TM/TC) • Persistent keys across resets • Store-and-forward tolerant • Authenticated & encrypted telemetry • Modular crypto engine support	• FSW security policy interface • Security Association (SA) orchestration • Key generation, activation, & rotation • Remote rekeying coordination • Operator SDLS monitoring & control • Role-based access control for security operations

FY25: AerLock v1 (App Layer) Development → **FY26:** AerLock v2 & Initial Security Manager → **FY27:** Full Security Manager & Constellation Support